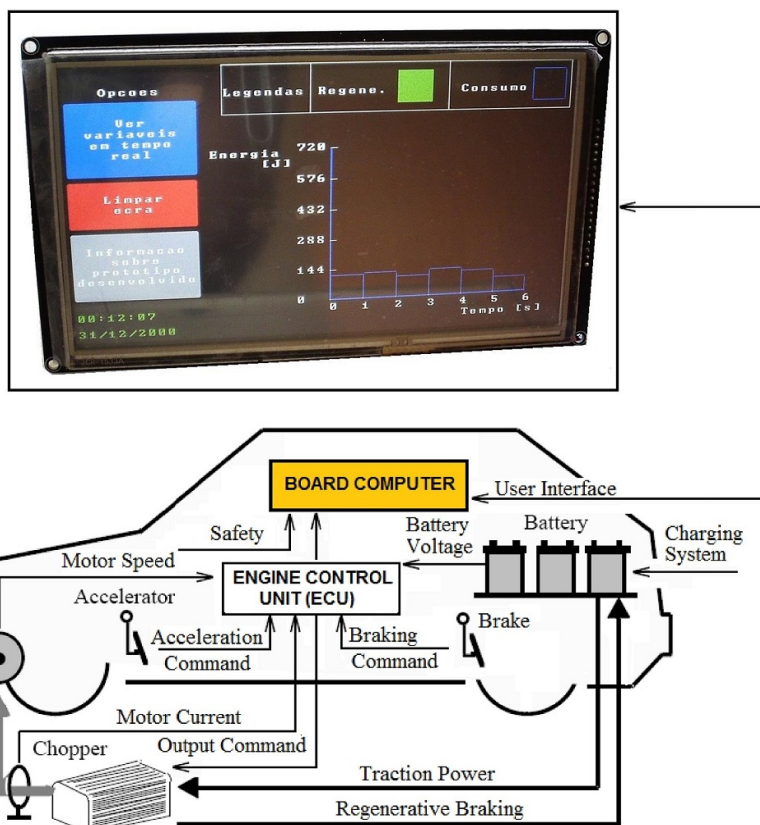


Ricardo Manuel
Guerra Pina

Desenvolvimento de um Carro Elétrico Puro: Instrumentação e Registos Energéticos

Pure Electric Car Development: Instrumentation and Energetic Registry





**Ricardo Manuel
Guerra Pina**

**Desenvolvimento de um Carro Elétrico Puro:
Instrumentação e Registos Energéticos
Pure Electric Car Development: Instrumentation
and Energetic Registry**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Automação Industrial, realizada sob a orientação científica do Doutor Rui Manuel Escadas Ramos Martins, Professor Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Aos meus Pais e Irmãos.

o júri / the jury

presidente / president

Prof. Doutor Pedro Nicolau Faria da Fonseca

Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Valter Filipe Miranda Castelão da Silva

Professor Adjunto, Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

Prof. Doutor Rui Manuel Escadas Ramos Martins

Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

**agradecimentos /
acknowledgements**

Agradeço ao Professor Doutor Rui Martins pela oportunidade cedida e auxílio técnico.

Ao Tiago Gonçalves pelo auxílio técnico e material cedido.

Agradeço igualmente aos meus colegas de curso pela companhia durante os últimos anos.

Palavras Chave

bateria, carro elétrico, comunicação, energia, instrumentação, Indústria Automóvel, On-board Diagnostics, regeneração, supervisão, Controller Area Network (CAN)

Resumo

Desde meados do século XX os avanços na indústria automóvel trazem a associação da eletrónica sendo esta cada vez mais necessária.

Com este trabalho pretende-se construir um computador de bordo de um veículo elétrico dando continuação a uma dissertação anterior, que consistiu em um Controlador de Motor de carro elétrico com capacidade de travagem regenerativa. Este Computador deve ser de baixo custo e é destinado à instrumentação, apresentação de balanços energéticos e funções básicas de diagnóstico do estado do veículo. Foi igualmente um objetivo desenvolver uma *Interface* a partir de um LCD para este Computador de Bordo. No modelo de Controlador de Motor disponível foram aplicadas alterações e adições na instrumentação de forma a conseguir um diagnóstico do veículo mais preciso e extenso.

Para o computador ter conhecimento do consumo e outros parâmetros relacionados com o motor e respetivo Controlador, foi necessário estabelecer uma comunicação entre ambas unidades. Antes de se implementar um protocolo de comunicação realizou-se uma pesquisa por protocolos usados na indústria automóvel, com o intuito de saber qual o mais apropriado para o presente trabalho. A tarefa seguinte consistiu em uma pesquisa por *hardware* com o qual desenvolver o Computador e sua *Interface*. Os balanços energéticos implicaram o desenvolvimento de métodos de cálculo, efetuados com os parâmetros transmitidos pelo Controlador de Motor.

No final da dissertação demonstra-se todas as funcionalidades do Computador de Bordo desenvolvido e como este é utilizado.

Keywords

battery, electric car, communication, energy, instrumentation, automotive industry, On-Board Diagnostics, regeneration, supervision, Controller Area Network (CAN)

Abstract

Since the mid-20th century the advances in the automotive industry have brought the association of the electronics being this one increasingly more required.

With this work it is intended to develop an On-Board Computer of an electric vehicle giving continuation to a previous dissertation, which consisted in an electric car Motor Controller with the regenerative braking ability. This computer should be low cost and is designed to the instrumentation, presentation of energetic balances and to basic diagnosis features of the vehicle state. It is also a goal to perform an Interface from a LCD for this On-Board Computer. In the available model of Motor Controller were applied changes and additions to the instrumentation in order to achieve a more precise and extensive diagnosis of the vehicle.

In order to the computer have knowledge about the consumption and other parameters related to the motor and respective Controller, it was necessary to implement a communication protocol between both units. Before implementing the communication it has been performed a research about protocols used in the automotive industry, with the aim to know which protocol is the most appropriate for current work. The following task consisted in a research for hardware with which to develop the Computer and its Interface. The energetic balances involved the development of calculation methods, made from the parameters transmitted by the Motor Controller.

At the end of the dissertation it is shown all the features of the developed computer and how it is used.

Conteúdo

| | | |
|----------|---|-----------|
| 1 | Introdução | 1 |
| 1.1 | Enquadramento | 1 |
| 1.2 | Objetivos | 2 |
| 2 | Estado da Arte | 3 |
| 2.1 | Breve história das aplicações eletrónicas nos carros | 3 |
| 2.2 | Protocolos de comunicação na indústria automóvel | 4 |
| 2.3 | <i>Controller Area Network</i> | 5 |
| 2.3.1 | Sinais no barramento e endereçamento | 5 |
| 2.3.2 | Acesso ao meio | 6 |
| 2.3.3 | Tipos de trama | 7 |
| 2.4 | <i>Local Interconnect Network</i> | 10 |
| 2.4.1 | Arquitetura | 11 |
| 2.5 | <i>FlexRay</i> | 11 |
| 2.5.1 | Arquitetura | 12 |
| 2.6 | <i>On-Board Diagnosis</i> | 13 |
| 2.6.1 | Modo de utilização do OBD-II | 15 |
| 2.6.2 | Equipamentos de diagnóstico | 16 |
| 2.7 | <i>Interface</i> sobre consumo nos veículos elétricos | 17 |
| 3 | Comunicação a desenvolver e principal <i>Hardware</i> | 19 |
| 3.1 | Comunicação a implementar e novo MCU do Controlador | 19 |
| 3.1.1 | Protocolos de comunicação usados na indústria automóvel | 19 |
| 3.1.2 | Fatores a ponderar na escolha de protocolo | 20 |
| 3.1.3 | Protocolo de comunicação a implementar | 20 |
| 3.1.4 | Novo MCU | 20 |
| 3.2 | <i>Hardware</i> de Computador de Bordo e sua <i>Interface</i> | 21 |
| 3.2.1 | 4D <i>Systems</i> - Picadillo - 35T | 21 |
| 3.2.2 | 4D <i>Systems</i> - uLCD - 70DT | 22 |
| 3.2.3 | SainSmart MEGA 2560 + TFT LCD de 7" + <i>Shield</i> TFT | 22 |
| 3.2.4 | SainSmart Due + TFT LCD de 7" + <i>Shield</i> TFT | 23 |

| | | |
|----------|---|-----------|
| 3.3 | Ambientes de desenvolvimento | 24 |
| 3.3.1 | dsPIC | 24 |
| 3.3.2 | Arduino | 25 |
| 3.4 | Sistema a desenvolver | 25 |
| 4 | Unidade de Controlo de motor existente e ajustes | 27 |
| 4.1 | Motor e alimentação da eletrónica | 27 |
| 4.2 | Controlo do motor BLDC | 28 |
| 4.3 | Métodos de <i>debug</i> | 30 |
| 4.4 | Interruptores e sinais de controlo | 31 |
| 4.5 | Instrumentação | 31 |
| 4.5.1 | Sensores de <i>Hall</i> para posição | 32 |
| 4.5.2 | Sensor de corrente CASR50 | 32 |
| 4.5.3 | Sensores de temperatura LM35DT | 35 |
| 4.5.4 | Amplificadores isolados AMC1200 | 35 |
| 5 | Desenvolvimento da comunicação CAN | 39 |
| 5.1 | <i>Transceiver</i> CAN | 39 |
| 5.2 | Identificação de parâmetros e configurações do protocolo | 39 |
| 5.3 | dsPIC33FJ64GS606 - Unidade de Controlo | 41 |
| 5.3.1 | Biblioteca CAN | 41 |
| 5.3.2 | <i>Hardware</i> relacionado com a comunicação | 42 |
| 5.3.3 | Envio, receção e processamento de mensagens CAN | 42 |
| 5.4 | Arduino Due - Computador de Bordo | 44 |
| 5.4.1 | Capacidades CAN do ATSAM3X8E e ligações elétricas | 44 |
| 5.4.2 | Biblioteca CAN | 46 |
| 5.4.3 | Envio, receção e processamento de mensagens CAN | 46 |
| 6 | Computador de Bordo - Balanços energéticos e respetivo registo | 49 |
| 6.1 | Inicialização do Computador de Bordo | 49 |
| 6.2 | Comunicação com cartão SD | 51 |
| 6.2.1 | <i>Pinout</i> e diferentes <i>interfaces</i> nos cartões SD | 51 |
| 6.2.2 | <i>Serial Peripheral Interface</i> | 52 |
| 6.2.3 | Protocolos SD | 55 |
| 6.2.4 | Protocolo utilizado | 56 |
| 6.2.5 | Ligações para cartão SD, LCD e painel táctil | 56 |
| 6.3 | <i>Real Time Clock</i> | 57 |
| 6.4 | Cálculo de consumos e regeneração energética | 58 |
| 6.5 | Registo nos ficheiros de texto | 60 |
| 7 | Computador de Bordo - <i>Interface</i> | 63 |

| | | |
|----------|---|-----------|
| 7.1 | Técnicas de desenvolvimento da componente visual da <i>Interface</i> | 63 |
| 7.2 | Ligações elétricas do LCD e respetivas funções | 64 |
| 7.3 | Desenvolvimento da componente tátil | 65 |
| 7.3.1 | <i>Hardware</i> | 65 |
| 7.3.2 | <i>Software</i> | 65 |
| 7.4 | Funções da <i>Interface</i> desenvolvida e botões tácteis | 67 |
| 7.4.1 | Teclado numérico para configuração de data e hora | 68 |
| 7.4.2 | Ver variáveis em tempo real | 69 |
| 7.4.3 | Ver registos recentes de consumo e regeneração de experiências associadas | 70 |
| 7.4.4 | Alertas | 72 |
| 7.4.5 | Informação acerca todo o protótipo desenvolvido | 73 |
| 7.4.6 | Limpar ecrã | 74 |
| 7.5 | Consumos energéticos de todo o sistema eletrónico | 74 |
| 8 | Conclusões e trabalho futuro | 77 |
| 8.1 | Conclusões | 77 |
| 8.2 | Trabalho futuro | 78 |
| | Bibliografia | 79 |
| | Anexos | 83 |
| A | Excerto da ficha de dados do AMC1200 | 84 |
| B | Excerto da ficha de dados XPT2046 | 86 |

Lista de Figuras

| | | |
|------|---|----|
| 1.1 | Diagrama de blocos do veículo elétrico a desenvolver [1]. | 2 |
| 2.1 | Protocolos aplicados num automóvel [2]. | 4 |
| 2.2 | Exemplo de sinais num barramento CAN [3]. | 6 |
| 2.3 | Trama de dados do CAN2.0A [3]. | 8 |
| 2.4 | Trama de dados do CAN2.0B, ou extendida [3]. | 9 |
| 2.5 | Trama no protocolo LIN [4]. | 11 |
| 2.6 | Trama do protocolo <i>FlexRay</i> [4]. | 12 |
| 2.7 | Conector OBD-II [5]. | 14 |
| 2.8 | ESItronic KTS 340 [6]. | 16 |
| 2.9 | Equipamento de OBD-II elementar [7]. | 16 |
| 2.10 | Ecrã de bordo com informação sobre consumo e regeneração energética [8]. . . . | 17 |
| 3.1 | Vistas frontais e traseiras do Picadillo - 35T [9]. | 21 |
| 3.2 | Vistas frontais e traseiras do uLCD - 70DT [10]. | 22 |
| 3.3 | LCD da SainSmart (em cima), <i>shield</i> (ao centro) e SainSmart MEGA2560 (em baixo) [11]. | 23 |
| 3.4 | LCD da SainSmart, <i>shield</i> e SainSmart Due [12]. | 24 |
| 3.5 | Diagrama de blocos do <i>hardware</i> | 26 |
| 4.1 | Esquema elétrico do Controlador de BLDC. | 29 |
| 4.2 | Circuito elétrico de um braço da ponte H utilizada. | 30 |
| 4.3 | Circuito da placa de medidas. | 31 |
| 4.4 | Sinal de saída do CASR50 em duas diferentes escalas temporais. | 34 |
| 4.5 | Divisor resistivo equivalente para acondicionamento das tensões das baterias. . . | 36 |
| 5.1 | Captura de trama CAN 2.0A. | 41 |
| 5.2 | Fluxograma transmissão de mensagens CAN no dsPIC33F. | 43 |
| 5.3 | Inserção de valor de parâmetros CAN em vetor de envio. | 44 |
| 5.4 | Ligações elétricas no Arduino Due. | 45 |
| 5.5 | Placa do <i>transceiver</i> CAN do Computador de Bordo. | 45 |
| 5.6 | Fluxograma de inicialização de parâmetros CAN. | 47 |
| 6.1 | Fluxograma da inicialização do Computador de Bordo. | 50 |

| | | |
|-----|--|----|
| 6.2 | Diagrama de pinos de um cartão SD [13]. | 51 |
| 6.3 | Topologia de ligações SPI [14]. | 53 |
| 6.4 | Sinais e tempos de amostragem de uma comunicação SPI com $CPOL = 0$ e $CPHA = 0$ [15]. | 53 |
| 6.5 | Esquema elétrico do <i>shield</i> LCD. | 57 |
| 6.6 | Fluxograma de cálculo de balanços energéticos e respetivo registo. | 59 |
| 6.7 | Fluxograma de gestão de ficheiros de texto e respetiva escrita. | 61 |
| 6.8 | Exemplo de registos energéticos em ficheiro. | 62 |
| 7.1 | <i>Interface</i> do <i>software</i> de conversão de imagens. | 64 |
| 7.2 | Fluxograma sobre leitura processamento de dados adquiridos pelo painel tátil. | 67 |
| 7.3 | Teclado matricial. | 68 |
| 7.4 | Menu de visualização de parâmetros em tempo real. | 69 |
| 7.5 | Menu de balanços energéticos, realizados durante uma fase de consumo. | 70 |
| 7.6 | Acoplamentos entre motores para ensaios de consumo e regeneração. | 71 |
| 7.7 | Gráfico adquirido durante travagem regenerativa. | 72 |
| 7.8 | Exemplo de alerta de temperatura. | 73 |
| 7.9 | Menu de informação sobre protótipo desenvolvido. | 74 |

Lista de Tabelas

| | | |
|-----|---|----|
| 2.1 | Protocolos de comunicação de existentes dentro da especificação OBD-II. | 14 |
| 5.1 | Parâmetros transmitidos pela rede CAN. | 40 |
| 6.1 | Funções dos pinos de cartão SD [16]. | 52 |
| 6.2 | Valores e significados dos parâmetros CPOL e CPHA [15]. | 54 |
| 6.3 | Diferentes modos de operação do SPI. | 54 |
| 6.4 | Estrutura de tramas de comando[16]. | 54 |
| 6.5 | Alguns comandos SD [16]. | 55 |
| 6.6 | Pacote de dados de uma comunicação com cartão SD no modo SPI [17]. | 55 |
| 7.1 | Sinais e alimentação LCD. | 65 |
| 7.2 | Correspondência entre botões e funções da <i>Interface</i> | 68 |
| 7.3 | Consumos energéticos da UC e Computador de Bordo. | 75 |

Lista de Acrónimos

| | |
|-------------|--|
| API | <i>Application Programming Interface</i> |
| BLDC | <i>Brushless Direct Current</i> |
| CAN | <i>Controller Area Network</i> |
| CARB | <i>California Air Resources Board</i> |
| CR | <i>Collision Resolution</i> |
| CRC | <i>Cyclic Redundant Check</i> |
| CSMA | <i>Carrier Sense Multi-Access</i> |
| DCR | <i>Deterministic Collision Resolution</i> |
| DLC | <i>Data Length Code</i> |
| ECAN | <i>Enhanced CAN</i> |
| ECU | <i>Electronic Control Unit</i> |
| EOF | <i>End of Frame</i> |
| FCEM | <i>Força Contra-Eletromotriz</i> |
| FIFO | <i>First In First Out</i> |
| GM | <i>General Motors</i> |
| ICSP | <i>In-Circuit Serial Programming</i> |
| ID | <i>Identificador</i> |
| IDE | <i>Integrated Development Environment</i> |
| ISO | <i>Organização Internacional de Normalização</i> |
| LCD | <i>Liquid Crystal Display</i> |
| LIN | <i>Local Interconnect Network</i> |
| MCU | <i>Microcontroller Unit</i> |
| MISO | <i>Master In Slave Out</i> |
| MOSI | <i>Master Out Slave In</i> |

MOST *Media Oriented Systems Transport*
OBD *On-Board Diagnosis*
OSI *Open Systems Interconnection*
RGB *Red Green Blue*
RTC *Real Time Clock*
RTR *Remote Transmission Request*
SAE *Society of Automotive Engineers*
SCI *Serial Communications Interface*
SD *Secure Digital*
SOF *Start of Frame*
SPI *Serial Peripheral Interface*
SRR *Substitute Remote Request*
TDMA *Time Division Multiple Access*
TFT *Thin Film Transistor*
TTCAN *Time-Triggered CAN*
UART *Universal Asynchronous Receiver/Transmitter*
UC *Unidade de Controlo*

Capítulo 1

Introdução

Neste capítulo introdutório será frisado o enquadramento justificando a importância deste trabalho e explicados quais os objetivos atingir.

1.1 Enquadramento

O crescente investimento, por parte de diferentes fabricantes de automóveis em veículos elétricos evidencia a importância atual deste tipo de veículo no mercado automóvel. Com este investimento, hoje em dia são fabricados carros com a capacidade de percorrerem distâncias superiores a 130 *km* com um único carregamento [18]. Continuando a ser uma grande desvantagem deste tipo de veículo, os tempos de carregamento completo (ou convencional) tendem a diminuir, existindo modelos que atualmente carreguem em cerca de 4 horas. No entanto existem modos de carregamento rápido tiram proveito apenas de parte da bateria, carregando cerca de 80 % desta em meia hora. Estes métodos por vezes exigem equipamento de carga externo [19]. Em Portugal existem mais de 460 postos de carregamento que cobrem todas as principais cidades [20].

Com as características anteriores atingidas, os veículos elétricos têm uma capacidade inédita de competir com os tradicionais veículos de combustão interna. Isto verifica-se pela crescente venda de veículos elétricos [21].

Outro facto relevante na indústria automóvel são os avanços na instrumentação e eletrónica na indústria automóvel desde meados do século *XX* [22] que cada vez trazem mais eficientes e práticas técnicas de diagnóstico do estado de um automóvel. Desde o início da década de 2000 com a introdução do *infotainment* automóvel [23], desenvolveram-se veículos, sendo exemplo o Toyota Prius [24], que consegue transmitir ao condutor os balanços energéticos, assim como outros dados relacionados com a instrumentação.

1.2 Objetivos

Com o presente trabalho pretende-se reconstruir o Controlador já existente de um motor de carro elétrico e aplicar-lhe certas melhorias e expansões nas quais se destaca um Computador de Bordo. Este Computador de Bordo de carro elétrico puro deve conter a respetiva *Interface* com o utilizador sendo esta realizada por intermédio de um LCD tátil. Este computador será focado na instrumentação em geral o que significa que a funcionalidade deste computador é registar e processar variáveis relacionadas com o motor e respetivo Controlador¹. Para concretizar os objetivos anteriores, será necessário, estabelecer uma comunicação entre ambas unidades e combinadas estas unidades formam um sistema *standalone*² que permite fazer o diagnóstico ao estado do motor e respetivo controlador.

O Controlador e *chopper*³ existentes são resultado da dissertação apresentada em 2013 “Projeto e construção de um Controlador para motor DC sem escovas” [25] e têm a capacidade de travagem regenerativa.

Exemplos de variáveis a transmitir são os valores do acelerador, travão, corrente elétrica no motor, corrente injetada nas baterias, velocidade de rotação do motor e temperatura do motor. Com o registo destas variáveis serão realizados estimativas e registos do consumo energético e da energia absorvida/recuperada pelo sistema de travagem regenerativa. A *Interface* desenvolvida deve ser capaz de informar o utilizador do consumo e regeneração energéticas dos últimos tempos.

Na figura 1.1 encontra-se o diagrama de blocos do protótipo de carro elétrico que se pretende obter, destacando o bloco mais relativo à presente dissertação. Na mesma figura a UC é designada *Engine Control Unit* (ECU), tratando-se de uma designação típica na indústria atribuída às unidades eletrónicas controladoras dos sistemas de potência.

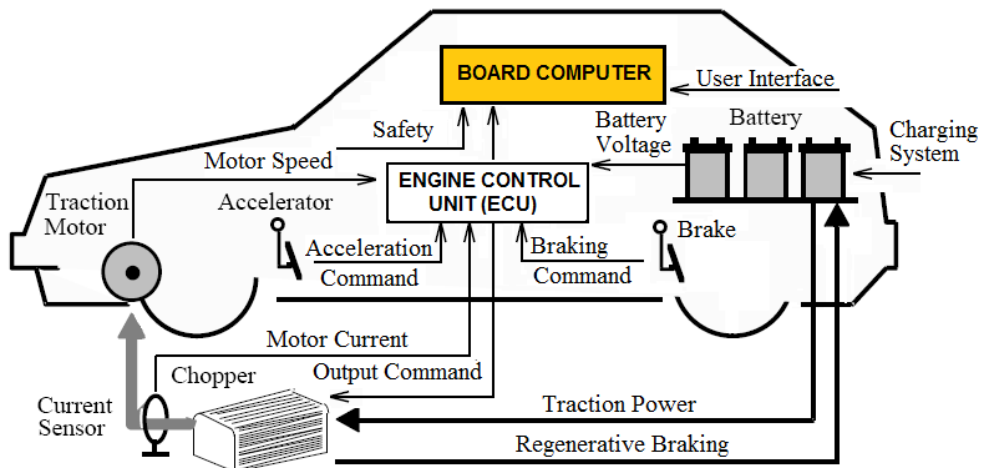


Figura 1.1: Diagrama de blocos do veículo elétrico a desenvolver [1].

¹Ou Unidade de Controlo (UC)

²Que não requer equipamento externo

³Conversor de tensão contínua fixa para tensão contínua variável, funciona como interruptor

Capítulo 2

Estado da Arte

Com o seguinte Estado da Arte faz-se a revisão sobre as aplicações eletrónicas na indústria automóvel ao longo dos anos. O principal aspeto tido em conta é a comunicação entre as várias unidades de controlo de um automóvel, o que significa que serão descritos quais os protocolos mais utilizados nesta indústria, de forma a que mais adiante nesta dissertação se defina qual protocolo de comunicação a implementar entre Controlador de motor e Computador de Bordo.

É realizado um estudo generalizado sobre os métodos recorrentes de diagnóstico automatizado.

2.1 Breve história das aplicações eletrónicas nos carros

A introdução da eletrónica na indústria automóvel surgiu, no final da década de 1960, com a necessidade gerir os processos do motor, resultando na introdução da injeção eletrónica de combustível. Desde esta última implementação de controlo eletrónico nos mecanismos dos automóveis, as aplicações eletrónicas foram-se expandindo tendo como principais marcos [23]:

- Década 1970: o controlo eletrónico influenciou vários processos no motor. Outras aplicações surgiram como o fecho de portas central e o *Cruise Control*. Este último sistema permite manter o veículo a uma velocidade previamente programada;
- Década 1980: foi introduzido o *Anti-lock Braking System* (ABS), que evita o bloqueio das rodas durante uma travagem. O ar condicionado começou a ser uma extra nos automóveis;
- Década 1990: no início desta década os *Airbags* tornam-se *standard* mais tarde, surgem sistemas de bloqueio de portas automático e desbloqueio sem chave. Um sistema notável surgido nesta época é o *Dynamic Stability Control*¹ (DSC);

¹Ou Controlo de Estabilidade Dinâmico

No estado da arte serão abordadas as principais características dos protocolos de comunicação acima referidos, com exceção ao MOST visto que as aplicações deste afastam-se das necessidades da corrente dissertação.

É de fazer referência aos seguintes protocolos que apesar de serem usados em menor escala que os anteriormente referidos, possuem uma ainda notável aplicação no mundo automóvel [2, 26]:

- *Time Triggered Protocol* (TTP) - Lançado em 1994, aparentava ser um protocolo promissor, no entanto apresentava menor viabilidade que o *FlexRay* para substituir o CAN;
- *Ethernet* - Protocolo a ser recentemente explorado na indústria automóvel;
- *Bluetooth* - Protocolo que permite dispositivos multimédia comunicarem sem fios;
- *ZigBee* - Protocolo de comunicação sem fios de baixo custo, aplicado a redes de sensores.

2.3 *Controller Area Network*

Desenvolvido a partir de 1983 pela empresa Robert Bosch, com intuito de interligar os sistemas eletrónicos nos automóveis com uma comunicação rápida, viável e com ligação física que resolve a questão da enorme quantidade de condutores exigidas pelos protocolos antecessores.

Em 1986 a Robert Bosch propôs oficialmente este protocolo definindo a comunicação ao nível físico e de ligação de dados, ou seja as camadas 1 e 2 do modelo *Open Systems Interconnection* (OSI), respetivamente. Notar que mais adiante será verificado que existem versões deste protocolo que definem camadas mais altas. No ano seguinte surgiram os primeiros circuitos integrados (controladores, por exemplo) produzidos pela Intel e pela Philips. Em 1993 o CAN foi normalizado com a ISO 11898 (que ao longo dos anos teve diferentes versões), e deste modo a sua aplicação expandiu-se ao ponto de se aplicar na automação em geral. Em 1992 fabricou-se o primeiro veículo a incorporar um barramento CAN.

2.3.1 Sinais no barramento e endereçamento

Trata-se de um protocolo de comunicação série assíncrono, pois as estações apenas se sincronizam no início de cada trama. Geralmente recorre a barramento diferencial a dois fios de modo a atribuir a este uma maior imunidade ao ruído. Estas linhas de transmissão são designadas CAN *High* e CAN *Low*.

Os sinais no barramento CAN podem ter dois diferentes níveis de tensão, designados “dominante” e “recessivo”. Estes correspondem aos valores lógicos 0 e 1 respetivamente². Os nomes “dominante” e “recessivo” aplicam-se visto que quando várias estações ocupam

²segundo as especificações CAN

simultaneamente o meio o sinal que prevalece na linha é o do *bit* dominante. É enviado o estado dominante quando a linha *CAN High* tem um valor superior à da linha *CAN Low*, enquanto o estado recessivo é enviado quando a linha *CAN Low* tem um valor superior ou igual (dependendo da especificação CAN) à linha *CAN High* assim como se ilustra na figura 2.2.

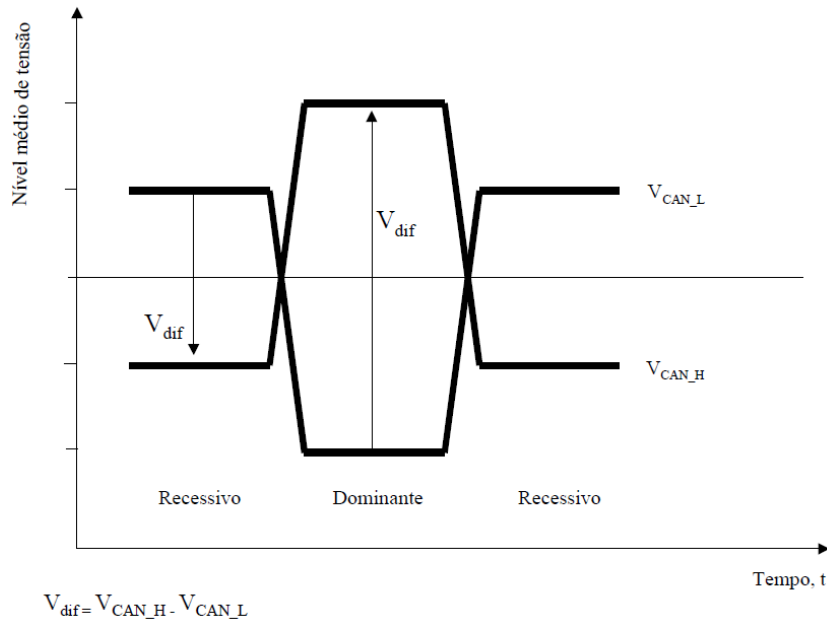


Figura 2.2: Exemplo de sinais num barramento CAN [3].

Este protocolo não recorre ao endereçamento convencional, para se distinguir as mensagens estas possuem um identificador (ID) de forma a que todos os nós que estejam a escutar o barramento possam decidir processar a mensagem ou não e, para além desta aplicação, definem a prioridade de acesso ao meio da mensagem. Na prática o identificador corresponde por exemplo ao parâmetro/variável a transmitir e quanto menor o valor do identificador maior é a sua prioridade.

A regra de inserção de *bits*³ aplica-se neste protocolo que permite a deteção de *flags*, contribui para a deteção de erros e ainda auxilia a sincronização entre estações. Esta regra indica que após os envio de uma sequência constituída por 5 *bits* com o mesmo valor é enviado um *bit* com valor diferente, que é retirado quando o recetor processa mensagem recebida.

Para conseguir uma *interface* entre controlador CAN e barramento diferencial a dois fios recorre-se a componentes designados *transceivers*.

2.3.2 Acesso ao meio

Trata-se de um rede multimestre pois vários nós podem requerer o acesso ao meio em qualquer instante, sem ter que ser numa resposta a um evento. Implementa o mecanismo de acesso

³Ou *Bit Stuffing*

ao meio *Carrier Sense Multi-Access / Deterministic Collision Resolution* (CSMA/DCR) em que as estações escutam o meio para saber se este está livre, este procedimento resume-se nos seguintes passos:

- Ao se transmitir o identificador de uma mensagem, todos os nós monitorizam o barramento;
- Se o transmissor enviar um *bit* recessivo e este observar o barramento com a tensão de um *bit* dominante, é detetada uma colisão e então o nó desiste da transmissão;
- O nó com objeto de menor identificador ganha o acesso ao meio e prossegue a transmissão.

Deste modo consegue-se uma arbitragem não destrutiva, isto é, a transmissão do objeto de menor identificador não sofre qualquer atraso. Lembrar que no início de cada trama de dados os nós sincronizam-se [3].

A máxima taxa de transmissão é de $1M\text{ bit/segundo}$ que é aplicável a barramentos com comprimento inferior a 40 m , tal como especificado na ISO 11898-2:2003. Outra norma a ter em conta é a SAE J2411 que permite o CAN e outros protocolos que implementem o *Carrier Sense Multiple Access/Collision Resolution* (CSMA/CR) comunicarem apenas por um fio, e neste caso o CAN suporta a velocidade máxima de 33 kbits/s [2, 27].

Time-Triggered CAN

Estes últimos aspetos verificam-se na versão do CAN orientada a eventos, no entanto foi desenvolvida uma diferente versão deste protocolo que implementa o *Time Division Multiple Access* (TDMA) em que o acesso ao meio é controlado por janelas de tempo predefinidas sendo este o *Time-Triggered CAN* (TTCAN), versão esta publicada na ISO 11898-4 de 2004 [28].

Neste tipo de comunicação existem dois tipos diferentes de janelas de tempo: janelas de tempo exclusivo e janelas de arbitragem. As janelas exclusivas são destinadas a mensagens transmitidas periodicamente, que não competem o acesso ao meio. Nas janelas de arbitragem são transmitidas as mensagens não ou pouco dependentes do tempo, e nestas janelas aplica-se o método não destrutivo de acesso ao meio referido anteriormente (CSMA). Neste protocolo que cumpre a camada de ligação do CAN é necessário um mestre que limite os ciclos de comunicação através de mensagens de referência [29]. É de fazer de referência ao protocolo *Flexible Time-Triggered on CAN* (FTT-CAN) desenvolvido na Universidade de Aveiro, em que a janela de tempo exclusivo é determinada dinamicamente pelo mestre dependendo do número de mensagens periódicas previstas conseguindo uma comunicação mais flexível.

2.3.3 Tipos de trama

Em 1991 foi lançada a especificação 2.0 do CAN permitindo identificadores de 29 bits (versão 2.0B) que cujas tramas são designadas “estendidas”, todavia as tramas com identi-

ficadores de 11 *bits* (versão 2.0A) continuaram a ser suportadas sendo designadas *standard*. De seguida são explicados os diferentes tipos de tramas e respetivo enquadramento.

Trama de dados

Na figura 2.3 demonstra-se a estrutura de uma trama de dados do CAN 2.0A.

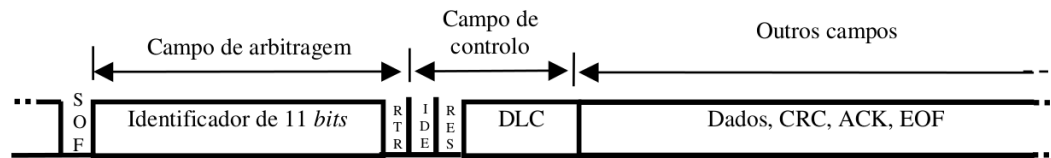


Figura 2.3: Trama de dados do CAN2.0A [3].

De seguida descrevem-se, por ordem de envio, os campos constituintes de uma trama de dados CAN:

- *Start of Frame*⁴ (SOF) - *Bit* dominante, informa que todos os nós devem-se sincronizar;
- Identificador - já explicada a sua função;
- *Remote Transmission Request*⁵ (RTR) - Recessivo quando se requer informação a outro nó, dominante quando se envia dados;
- *Identifier Extension*⁶ (IDE) - *Bit* dominante para mensagens *standard*, recessivo para estendidas;
- Reservado - *Bit* reservado;
- *Data Length Code* (DLC) - Código do número de *bytes* a ser transmitido;
- Dados - Campo com comprimento variável podendo ir até 8 *bytes*;
- *Cyclic Redundant Check* (CRC) - Campo acrescentado para deteção de erros, tem comprimento de 15 *bits* mais 1 *bit* de delimitador;
- *Acknowledgement*⁷ - Cada remetente envia o *bit* de reconhecimento com o valor recessivo e cada recetor, caso detete uma mensagem livre de erros, sobrescreve este *bit* para dominante. Este campo é constituído por 2 *bits*, o *bit* de reconhecimento e o delimitador;
- *End of Frame*⁸ (EOF) - Campo de delimitador de toda trama, é constituído por 7 *bits* recessivos, infringindo assim a regra de inserção de *bits*;

⁴Início de trama

⁵Pedido de transmissão remota

⁶Extensão de identificador

⁷Reconhecimento

⁸Fim da Trama

- Intermissão - Durante este campo de 3 *bits*, nenhuma estação pode ocupar a estação a não ser para enviar uma trama de sobrecarga, tipo este de trama que será explicado mais adiante. Após a intermissão o barramento fica desocupado e então as estações podem dar início à transmissão de tramas de dados e remotas [3, 30].

As tramas estendidas e *standard* não variam apenas no comprimento do identificador, como se pode verificar na figura 2.4.

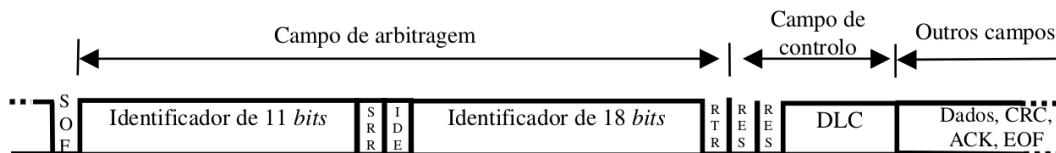


Figura 2.4: Trama de dados do CAN2.0B, ou estendida [3].

Como já referido o *bit* IDE é recessivo, e é acrescentado o *bit Substitute Remote Request*⁹ (SRR), que agora tem a função de indicar o tipo de trama em questão. É ainda incluído um novo *bit* reservado.

Trama remota

Neste protocolo, a transmissão de dados pode ser devida a um pedido. Para tal ocorrer é necessário que um nó envie uma trama remota que contem um formato semelhante a uma trama de dados com a diferença de não conter campo de dados, para além de o *bit* RTR ser recessivo tal como referido anteriormente.

Gestão de erros e tramas associadas

Se numa rede CAN detetar-se um erro de comunicação, é enviada uma trama de erros avisando todos os elementos da rede acerca do erro. Esta trama é constituída pelo campo de erro e o campo delimitador.

O primeiro não cumpre a regra de inserção de *bits* visto consistir numa sequência de *bits* como o mesmo valor. Este mesmo campo tem como base conjunto de *flags* de erro, cada uma enviada por um nó que deteta o erro. Notar que cada *flag* tem que ter no mínimo 6 *bits* para assim violar a regra de inserção de *bits*.

As estações podem ser ativas ou passivas ao erro assim como as respetivas *flags* de erro. Cada estação possui em si um contador de erros de transmissão e um contador de erros de receção. Sempre que é detetado um erro de comunicação cada estação incrementa o contador relacionado à posição na comunicação (emissor ou recetor). Geralmente o contador de erros de transmissão é incrementado mais rapidamente (maior valor incrementado de cada vez), devido à maior probabilidade de o emissor ser o causador do erro. Enquanto as estações

⁹Pedido remoto de substituição

tiverem ambos contadores com valores inferiores a 127, estas estações comportam-se como estações “ativas ao erro” de maneira que as sua *flags* de erro consistem em séries de *bits* dominantes que destroem eventuais mensagens em transmissão. Caso um dos contadores de erros ultrapasse o limite a estação entra no estado “passivo ao erro” o que significa que não causará interferências em mensagens que circulem no barramento. Quando o contador de erros de transmissão ultrapassa o valor 255 a estação abandona a comunicação entrando no modo “desligado do barramento”¹⁰ e após a deteção de várias sequências de *bits* recessivos (barramento livre) os contadores de erro são reiniciados e a estação volta ao estado “passivo ao erro”.

O segundo campo de uma trama de erro é composto por 8 *bits* recessivos e serve para indicar o fim da trama de erro [30].

Trama de sobrecarga

Este tipo de trama é enviado logo após uma trama de dados, ou seja durante a intermissão. O propósito desta trama é evitar erros na receção, causando um atraso extra no envio da próxima trama de dados ou remota de forma a que o nó emissor da trama de sobrecarga tenha tempo para processar a mensagem atual.

Estas tramas são constituídas por dois campos, *flags* de sobrecarga que consiste em 6 *bits* dominantes (constituindo assim uma *flag* de erro ativa) e o delimitador de sobrecarga que consiste em 8 *bits* recessivos [3, 30].

2.4 Local Interconnect Network

O LIN foi desenvolvido a partir de 1998 pelo consórcio LIN constituído por companhias como a Audi, BMW, Volkswagen, e Motorola com a finalidade de permitir uma comunicação de baixo custo para aplicações de baixa complexidade. No ano 2000 deixou de ser um protocolo proprietário tendo sido normalizado abertamente pela LIN 1.1. Foi projetado para ser aplicado na indústria automóvel com a finalidade de ser um complemento ao CAN em aplicações que exigissem menores larguras de banda e menores capacidades de deteção e correção de erro.

Na figura 2.1 estão representadas as aplicações típicas do LIN como o controlo dos trincos das portas, controlo orientação de espelhos, limpa-pára-brisas e leitura de sensores de chuva. Nestas aplicações o LIN tem uma melhor relação custo/eficiência em comparação ao CAN que possui maiores custos de *hardware* e *software*.

¹⁰ou *bus off*

2.4.1 Arquitetura

Uma rede LIN é constituída por um único mestre e vários escravos num máximo de 16 nós, em que é sempre o mestre a iniciar a comunicação formando uma topologia *Master-Slave*. É um protocolo de comunicação série e assíncrono. Cada trama é constituída por um cabeçalho e uma resposta. A ligação a nível físico corresponde ao *hardware* de uma *Universal Asynchronous Receiver/Transmitter Serial Communications Interface* (UART/SCI) habitual dos microcontroladores (MCUs). Como já mencionado, geralmente o LIN é usado como uma sub-rede do CAN.

Para iniciar uma comunicação o mestre envia o cabeçalho, na sequência caso pretenda enviar dados para o escravo o mestre envia a parte da resposta enquanto se o mestre pedir dados ao escravo este último envia a parte da resposta. Não existe comunicação direta entre escravos, todavia os pedidos do mestre podem ser usados para se realizar comunicações entre escravos visto todos os nós estarem a escutar o barramento.

Tal como o CAN, o LIN é orientado a objetos e então o cabeçalho contém um identificador que define a trama e o seu conteúdo. Outro campo existente no cabeçalho é o *byte* de sincronização, que permite aos escravo saberem qual a velocidade de transmissão ao mesmo tempo que se sincronizam. Isto permite ainda que os MCUs escravos terem osciladores de baixo custo. Outro aspeto interessante é a possibilidade de o mestre enviar tramas para induzir o modo *sleep* em qualquer nó e para reverter este modo qualquer nó pode enviar a trama com o sinal de acordar.

A verificação de erros é feita por *checksum* e verificação de paridade e não é implementado nenhum método de arbitragem o que contribui para uma menor complexidade de implementação. É uma rede *Time-Triggered* [2, 31, 32].

A figura 2.5 mostra um exemplo de trama LIN em que pode haver contribuições do mestre e escravo na mesma.

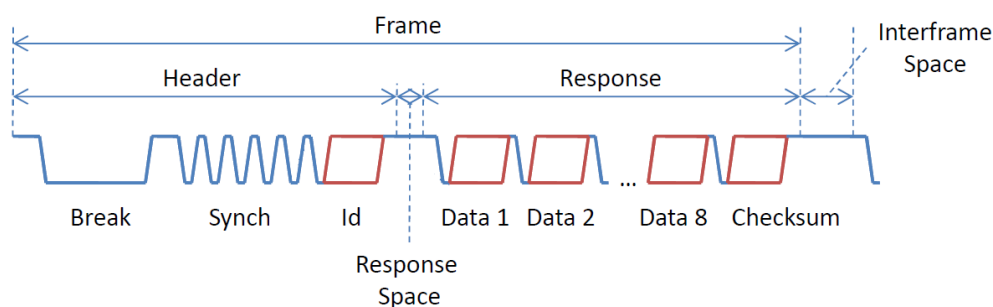


Figura 2.5: Trama no protocolo LIN [4].

2.5 FlexRay

No final da década de 1990 alguns fabricantes automóveis, concluíram que as exigências das aplicações eletrónicas futuras não seriam cumpridas com nenhum dos protocolos automó-

veis correntes. Deste modo o consórcio formado pelas companhias BMW, DaimlerChrysler, General Motors, Motorola, Philips, Volkswagen, e Robert Bosch desenvolveram este protocolo a partir do ano 2000. A primeira implementação para fins comerciais deste protocolo foi em 2006, para controlar a suspensão do BMW X5. Em 2009 o consórcio terminou e o protocolo foi publicado pela ISO 17458 de 2013. Este protocolo é considerado o futuro substituto do atual líder CAN [33].

2.5.1 Arquitetura

Com uma largura de banda 20 vezes superior ao CAN, consegue servir de meio de comunicação aos sistemas de controlo que requerem uma grande sincronização (o que traz segurança num automóvel) entre si para além permitir um viável atendimento a um maior número de nós. A velocidade máxima admissível é de 80 M bits/s e várias topologias de rede são permitidas sendo estas o barramento convencional, em estrela e combinação dos anteriores. O meio físico é o par trançado (tensões diferenciais) ou a fibra ótica. Uma característica inovadora é possibilidade de ligar cada nó a dois outros nós causando uma redundância para maior deteção de erros.

O protocolo *FlexRay* é um protocolo orientado a objetos, ou seja, as estações não têm endereço. As tramas enviadas são divididas em 3 secções principais: cabeçalho, dados (comprimento variável tal como no CAN) e fim de trama (CRC). A estrutura das tramas é demonstrada mais ao pormenor pela figura 2.6, verifica-se a presença de vários campos CRC.

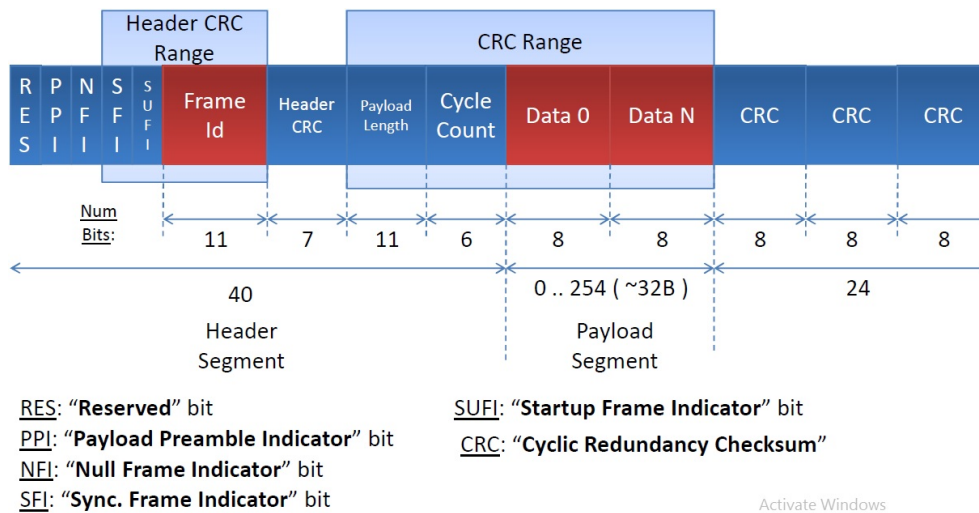


Figura 2.6: Trama do protocolo *FlexRay* [4].

O acesso ao meio pode ser controlado por tempo ou por eventos, isto é *Time-Triggered* ou *Event-Triggered* respetivamente. Ao contrário do CAN este protocolo apenas implementa tramas de dados. A versão *Time-Triggered* implementa o método TDMA e ainda algumas técnicas particulares [32, 2].

2.6 *On-Board Diagnosis*

Com os acontecimentos da secção 2.1, várias companhias realizaram sistemas computadorizados que permitem controlar as emissões e consumo estado do veículo. O primeiro destes sistemas foi desenvolvido pela Volkswagen nos seus modelos com injeção eletrónica de combustível, no final da década de 1960 com a principal finalidade de controlar as emissões poluentes que causavam problemas nos EUA como *smogs*. Este último processo exigiu uma *Electronic Control Unit* (ECU).

Com o crescimento dos sistemas eletrónicos no controlo do diferentes processos de um automóvel, como a injeção de combustível, circulação vapores no cárter e gases nos escape, em 1980 a General Motors (GM) produziu o *Assembly Line Diagnostic Link* (ALDL) que lia códigos de erro inicialmente a uma taxa (*baud*) de 160 *bits/segundo*, taxa esta que foi aumentada para 8192 *bits/segundo* em 1986 pela GM. Este sistema/protocolo foi adotado por vários fabricantes, que cada um por si fez as suas adaptações, o que fez por exemplo com que as fichas de ligação fossem diferentes (*pinouts* diferentes). Estas incompatibilidades levam a SAE, no ano de 1988, a recomendar uma ficha de ligação *standard*/padrão, assim como um protocolo de ligação padrão.

Devido à grande formação de *smogs*¹¹ no estado da Califórnia com o objetivo de reduzir as emissões de CO_2 , em 1991 a *California Air Resources Board* exige todos os veículos vendidos tenham certas capacidades fundamentais de *On-Board Diagnosis* (OBD), embora não exija uma ficha de ligação padrão nem um protocolo na camada de ligação específico. Esta última exigência/especificação inicialmente foi denominada OBD, que mais tarde mudou para OBD-I devido à especificação OBD-II de 1996 que cumpriu as recomendações da SAE anteriormente referidas. O sistema OBD-II não se aplicou apenas nos EUA visto que em 1998 a União Europeia ter decidido que todos os veículos a gasolina comercializados a partir de 2001 suportariam a mesma exigência, e, o mesmo se aplicou aos veículos a diesel a partir de 2004[22]. A regulamentação que se aplicou na Europa designou-se *European On-Board Diagnosis*.

Na figura 2.7 encontra-se o aspeto físico de um conector OBD-II e a legenda com as funções de cada pino.

¹¹Nevoeiros poluídos

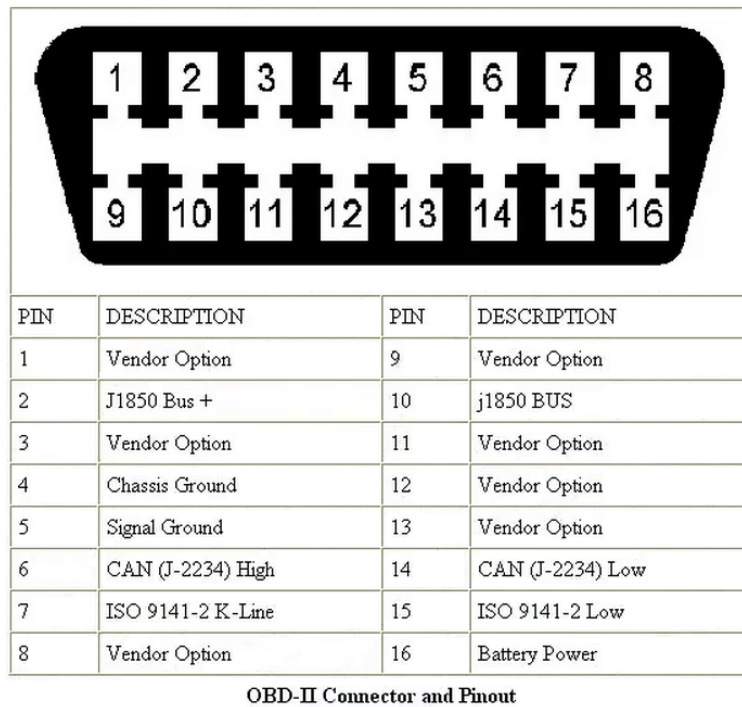


Figura 2.7: Conector OBD-II [5].

Pela figura anterior, concluí-se que o OBD-II não suporta apenas um protocolo comunicação, visto os fabricantes terem diferentes protocolos que recorrem a diferentes pares do conector. Deste modo, no total dentro desta especificação são usados os 6 diferentes protocolos registados na tabela 2.1.

Tabela 2.1: Protocolos de comunicação de existentes dentro da especificação OBD-II.

| Par (Ligação Física) | Protocolos | |
|----------------------------|--|---|
| J1850 | SAE J1850 VPW (<i>Variable Pulse Width Modulation</i>) | SAE J1850 PWM (<i>Pulse Width Modulation</i>) |
| ISO | ISO 9141 | ISO 4230 (por vezes conhecido por KWP 2000) |
| CAN | CAN de acordo com a ISO 15765 <i>standard</i> ou <i>extended</i> | |

Apesar da menor variedade de protocolos, continuaram a ocorrer complicações na análise de veículos dos vários fabricantes devido à incoerência de protocolos. Deste modo, nos EUA foi definido que todos veículos fabricados a partir de 2008 suportariam o protocolo CAN de acordo com a ISO 15765 – 4, que define as camadas 3 (rede) e 4 (transporte) [22, 34]. Verifica-se maioria dos fabricantes optam por usar o protocolo anterior desde 2006[5].

2.6.1 Modo de utilização do OBD-II

Independentemente do protocolo de comunicação a usar, no OBD-II a informação é retirada da *Engine Control Unit* (ECU) do veículo ao se enviar um de pedido. Estes códigos de pedido são designados *Parameter Identification* (PID), em que alguns PIDs são definidos pela norma SAE J/1979 e outros mais são definidos pelo fabricante de cada veículo [35].

Um aspeto importante a ter em conta são 10 diferentes modos de operação [36]:

1. Neste modo pode-se adquirir os parâmetros obtidos pelos sensores instalados no veículo, enviando para a ECU do veículo os respetivos PIDs;
2. Devolve os dados registados quando ocorreu uma falha;
3. Mostra códigos padrão/*standard* dos problemas de diagnóstico. Existem quatro diferentes categorias, distinguidas pelo primeiro carácter retornado (com uma codificação própria) pelo ECU:
 - P: Relacionado com elementos de potência, por exemplo motor, caixa de velocidades;
 - C: *Chassis*;
 - B: *Body* (carroçaria);
 - U: Rede de comunicações.
4. Limpa os códigos de problemas de diagnóstico registados. Se o problema se mantiver o código do problema será registado novamente;
5. Adquire medições dos sensores de oxigénio/*lambda*. Já não se aplica a veículos com ECUs com o protocolo CAN, e então é substituído pelo método 6;
6. Retorna dados dos sensores de oxigénio/*lambda* e outros processos de monitorização com funcionamento contínuo ou não;
7. Mostra códigos de problemas pendentes, que surgiram durante o último ciclo de condução. A resposta tem o mesmo forma que no modo 3;
8. Permite o controlo de um componente, processo que esteja relacionado o sistema OBD. Este modo é também apelidado teste de atuadores;
9. Retorna informação do veículo como o seu número de identificação, valores de calibração;
10. Por vezes conhecido por modo A (caso se recorra à numeração hexadecimal). Devolve códigos de falhas permanentes, que não eliminadas pelo modo 4. A CARB recomenda que os códigos de falhas que comandam luzes indicadoras devem ser permanentes.

Os fabricantes não são obrigados a disponibilizar todos os modos anteriores, podendo até desenvolver os seus modos desde que tenham um código acima de 9.

2.6.2 Equipamentos de diagnóstico

Certas companhias desenvolveram equipamentos com capacidades que vão além das requeridas OBD-II, por exemplo elaborar relatório de avarias, mencionar sugestões de reparação, *Interfaces* com diagramas que indicam a localização de avaria. Um exemplo de equipamento *standalone* multi-marcas¹² é o ESItronic KTS 340 da Bosch, em que o aspeto físico mostra-se na figura 2.8.



Figura 2.8: ESItronic KTS 340 [6].

Notar que o equipamento anterior corre o *software* de diagnóstico ESI[tronic] que pode ser corrido igualmente em um PC [37]. O preço destes equipamentos e respetivo *software* pode ultrapassar os 1000 €.

No mercado encontram-se equipamentos mais simples e para serem utilizados pelo condutor, que possuem as capacidade elementares de OBD-II com preços mais reduzidos, a figura 2.9 ilustra um exemplo deste equipamento.



Figura 2.9: Equipamento de OBD-II elementar [7].

¹² Certos fabricantes de automóveis desenvolvem os próprios equipamentos

2.7 *Interface* sobre consumo nos veículos elétricos

Naturalmente que os fabricantes, estabelecem diferentes formas de informar o utilizador sobre o estado de carga da bateria. Analogamente à indicação do nível de combustível num veículo de combustão interna, os veículos elétricos têm no seu painel de instrumentos indicadores de por exemplo percentagem de carga na bateria e potência máxima disponível.

Para além desses parâmetros por vezes é disponibilizada mais informação sobre o consumo energético. Certo veículos calculam o consumo médio dos últimos minutos e assim como a energia recuperada pela travagem regenerativa, que é o exemplo do Toyota Prius. Neste caso, estes dados não são ilustrados no painel de instrumentos diante do condutor, mas sim num ecrã que se encontra no centro do veículo (um exemplo de *infotainment*), como ilustrado na figura 2.10.



Figura 2.10: Ecrã de bordo com informação sobre consumo e regeneração energética [8].

Capítulo 3

Comunicação a desenvolver e principal *Hardware*

Neste capítulo será definido qual o protocolo a implementar, referido e justificado o novo MCU do Controlador do motor BLDC. É igualmente demonstrada a pesquisa por *hardware* com o qual desenvolver o Computador de Bordo. No final do capítulo é ilustrado um diagrama de blocos do *hardware* de todo sistema.

3.1 Comunicação a implementar e novo MCU do Controlador

A necessidade de estabelecer a comunicação entre Controlador de motor e Computador de Bordo, exigiu um estudo (demonstrado anteriormente) dos diferentes protocolos de comunicação para assim decidir qual o mais indicado para a aplicação pretendida. Verifica-se que o modelo de MCU previamente implementado nesta UC, o dsPIC33FJ16GS402, tem que ser substituído visto ter maioria dos seus pinos ocupados. A necessidade de usar um MCU com mais pinos existe devido à implementação de uma comunicação e para possibilitar eventuais expansões à UC, como por exemplo inserção de novos sensores.

3.1.1 Protocolos de comunicação usados na indústria automóvel

Como mencionado na secção 2.2, os protocolos que definem a comunicação, ao nível físico e de ligação de dados¹, mais usados na indústria automóvel são o CAN (o mais utilizado) e o LIN. O *FlexRay*, foi descartado devido ao reduzido número de MCUs que suportam este protocolo, por não serem necessárias taxas de transmissão tão elevadas e pela complexa implementação.

¹camadas 7 e 6 do modelo OSI, respetivamente

3.1.2 Fatores a ponderar na escolha de protocolo

Aplicação

Como já referido anteriormente, o sistema que se pretende desenvolver a partir de um MCU dsPIC trata-se de um Controlador de motor BLDC para ser implementado num carro elétrico, o que significa que as mensagens a transmitir consistem em valores obtidos por intermédio de sensores, como por exemplo temperaturas, correntes elétricas, etc. É necessário que na rede implementada ambas estações tenham a possibilidade de iniciar a comunicação. Deste modo é preciso reprogramar o MCU do Controlador em questão e assim estabelecer uma comunicação.

Família do MCU

A UC disponível é centrada em MCU da família de MCUs dsPIC33F, fabricados pela Microchip. Os MCUs desta família são igualmente considerados *Digital Signal Controller* (DSC), tratando-se de *chips* que unem as capacidades dos *Digital Signal Processor* (DSP) e dos MCU em geral.

O novo MCU deve ser da mesma série de forma a compatibilizar ao máximo o código já desenvolvido. Para cumprir esta última condição o protocolo a definir deve ser compatível com a família de MCUs utilizada.

3.1.3 Protocolo de comunicação a implementar

O protocolo escolhido para o presente trabalho é o CAN, porque para além de ter sido desenvolvido para a aplicação anterior, permite altas velocidades, imunidade ao ruído eletromagnético, baixa complexidade de condutores elétricos e facilidade de introdução de novas estações [3]. Outro fator tido em conta é quantidade de MCUs da série requerida que possuem suporte CAN.

Apesar dentro da mesma família haver MCUs com suporte de LIN, este não é escolhido visto este protocolo não ser indicado para interligar unidades da complexidade em questão e por não suportar comunicação multimestre.

A comunicação de entre Computador de Bordo e Controlador será abordada com mais detalhe no capítulo 5.

3.1.4 Novo MCU

Como já referido é necessário substituir o MCU do Controlador do motor existente. Realizou-se uma pesquisa por MCUs da mesma série, que tenham os mesmos periféricos como ADC e PWM em pares e de alta velocidade e que sejam compatíveis com o código fornecido do Controlador do motor. Outro requisito é naturalmente o suporte de comunicação CAN.

O código fornecido foi testado nos MCUs dsPIC33EP32GP504, dsPIC33FJ64MC506 e no dsPIC33FJ64GS606. O modelo escolhido é o dsPIC33FJ64GS606, visto ser o que exigiu menores alterações para se compilar o código existente.

3.2 *Hardware de Computador de Bordo e sua Interface*

O desenvolvimento de um Computador de Bordo exige uma *Interface* gráfica e uma unidade de processamento. No mercado encontra-se vários módulos constituídos por LCD e unidade de processamento gráficas baseadas em MCU. Após várias pesquisas, descobriu-se vários fabricantes de módulos gráficos baseado em MCUs. De seguida enunciam-se diferentes fabricantes e exemplos dos seus módulos gráficos.

3.2.1 4D *Systems* - Picadillo - 35T

O *Picadillo-35T* é um módulo centrado no MCU da Microchip PIC32MX795F512L possuindo um LCD-TFT de 3.5" embebido com resolução 320 X 480 e com painel tátil resistivo e contem ainda um altifalante. Permite a consulta armazenamento de conteúdos multimédia de dados em geral num cartão MicroSD. O seu aspeto visual apresenta-se na figura 3.1.



Figura 3.1: Vistas frontais e traseiras do Picadillo - 35T [9].

Este módulo foi desenvolvido para ser programado pelo *Integrated Development Environment*² (IDE) *Universal Embedded Computing IDE* (UECIDE), *software* este desenvolvido pela *Majenko Technologies* sendo também programável pelo MPLAB X da Microchip e pelo *Multi Platform IDE* (MPIDE) da chipKIT. Este módulo quando programado pelo MPIDE ou pelo UECIDE, pode-se recorrer a diversas bibliotecas fornecidas pelos IDEs, para além de certas bibliotecas do Arduino compatíveis [9].

²Ambiente de Desenvolvimento Integrado

3.2.2 4D Systems - uLCD - 70DT

O uLCD-70DT é destinado a ser trabalhado com o Arduino. Contém um LCD-TFT de 7", com resolução 800 X 480 e com painel tátil resistivo assim como o processador gráfico DIABLO16 desenvolvido igualmente pela 4D Systems. Tal como o módulo anterior, tem suporte para cartões MicroSD e ainda um altifalante. Este módulo ilustra-se na figura 3.2.

Pode ser configurado pelo IDE *Workshop4*, da 4D Systems podendo receber pela porta série comandos provenientes de uma unidade externa, para desenhar formas elementares. Pode também ser programado como um sistema autónomo pelo mesmo IDE, de uma forma mais complexa que elimina as limitações do método anterior[10].

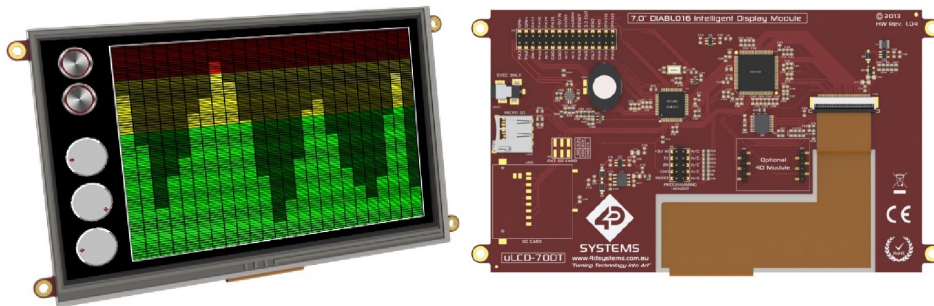


Figura 3.2: Vistas frontais e traseiras do uLCD - 70DT [10].

3.2.3 SainSmart MEGA 2560 + TFT LCD de 7" + *Shield* TFT

Este módulo por si incorpora um módulo TFT LCD de 7" com resolução 800 X 480, com painel tátil resistivo e suporte para cartões *Secure Digital* (SD). É fabricado e vendido pela SainSmart.

No mesmo módulo incorpora-se também uma placa de MCU SainSmart MEGA 2560, placa esta um "clone" do Arduino Mega 2560. Esta mesma placa é centrada no MCU ATmega2560. A interligação entre módulo LCD é realizada por um *shield*³ que incorpora um suporte para cartões SD, incluído no conjunto [11]. Na figura 3.3 mostra-se o aspeto físico de todo o módulo.

A maior vantagem deste módulo é o seu reduzido custo face aos modelos anteriores. Outra vantagem significativa são as dimensões do LCD, enquanto que a desvantagem mais significativa é que o MCU incluído na placa não contém qualquer periférico CAN.

³Elemento de ligação, neste caso entre LCD e MCU



Figura 3.3: LCD da SainSmart (em cima), *shield* (ao centro) e SainSmart MEGA2560 (em baixo) [11].

3.2.4 SainSmart Due + TFT LCD de 7" + *Shield* TFT

Tendo o mesmo fabricante que o módulo anterior, este conjunto inclui um módulo TFT-LCD igual ao incluído no conjunto anterior e um *shield* de ligação entre LCD e MCU. Inclui a placa de MCU centrada no Atmel SAM3X8E. O *shield* inclui um suporte para cartão SD, para comunicar com o MCU via *Serial Peripheral Interface* (SPI).

Esta última placa trata-se de um “clone” do Arduino Due. Visto o módulo LCD ser o mesmo, as bibliotecas destinadas a este módulo são as mesmas e tem a vantagem de o MCU suportar comunicações CAN [12]. Esta última vantagem em conjunto com o preço mais reduzido que dos conjuntos das subsecções 3.2.1 e 3.2.2 justificou o uso de módulo na presente dissertação. Outro fator tido em conta são as dimensões do LCD, sendo de maior dimensões oferece maior conforto na utilização face a um LCD com as dimensões do conjunto Picadillo -35T.

Na figura 3.4 os principais constituintes do conjunto. Apesar de não se tratar de um Arduino genuíno este módulo de MCU, em toda a restante dissertação será designado Arduino Due.

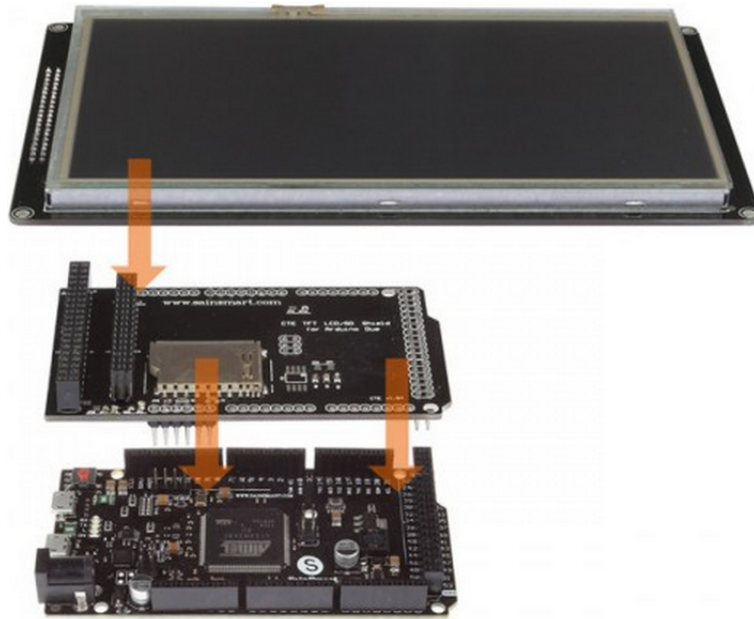


Figura 3.4: LCD da SainSmart, *shield* e SainSmart Due [12].

O módulo TFT LCD é equipado com controlador TFT LCD SSD1963 e com o controlador de painel tátil XPT2046 a 4 fios. A comunicação entre MCU e controlador LCD é realizada pela comunicação/*interface* paralela 8080 de 16 *bits* [38], enquanto que a comunicação entre MCU e controlador do painel tátil é por SPI[39].

O fabricante deste módulo TFT LCD não fornece nenhum IDE, embora indique certas bibliotecas compatíveis com o Arduino e assim como projetos de exemplo. As bibliotecas em questão tratam-se da UTFT (*Universal* TFT) para controlo do ecrã e da UTouch para utilização do painel tátil, desenvolvidas pela Rinky-Dink Eletronics [40]. As capacidades destas bibliotecas serão abordadas mais detalhadamente no capítulo 7, onde se aborda o trabalho realizado sobre o Computador de Bordo e respetiva *Interface*.

3.3 Ambientes de desenvolvimento

Nesta secção serão abordados os IDEs utilizados tanto para o Arduino (Computador de Bordo) e para o dsPIC33F (UC).

3.3.1 dsPIC

O *Integrated Development Environment* (IDE) para programar o dsPIC33FJ64GS606, é o *software* oficial do fabricante do MCU intitulado MPLAB X. Este *software* permite a criação e edição de projetos e suporte a compiladores. O compilador usado na programação do MCU dsPIC33F foi o XC16, destinado à linguagem C.

Outra capacidade deste IDE é a programação dos MCUs. No presente trabalho a programação do dsPIC33F foi realizada pelo método *In-Circuit Serial Programming* (ICSP). Para

além do carregamento de *firmware* para o MCU, possibilita a leitura de *firmware* contido no MCU e ainda é possível desprogramar. Este método requer um programador externo, e o utilizado foi o PICKit3, fabricado pela Microchip.

As ligações destinadas a este método de programação são:

- *Master Clear* (MCLR) - Para reiniciar o MCU;
- VDD - Tensão de alimentação;
- VSS - Massa de alimentação ou 0 V;
- *Programmer Data* (PGD) - Linha de envio de dados;
- *Programmer Clock* (PGC) - Linha de sinal de relógio de sincronismo.

3.3.2 Arduino

O IDE utilizado é o oficial do Arduino, a linguagem de programação deste IDE é o C++ embora o acesso a bibliotecas *standard* desta linguagem seja limitado. Este IDE permite acesso a diversas bibliotecas e *device drivers* que permitem facilmente por exemplo conversões analógicas digitais, habilitar interrupções externas etc.

Para programação do Arduino Due, tirou-se proveito do facto de este ser comercializado, já pré-programado com um *bootloader*. *Bootloader* é um *firmware* alojado na *flash* que permite uma programação do MCU por ambas portas série desta placa⁴, evitando assim o recurso a um programador externo. A desvantagem de usar um *bootloader* é o consumo de memória *flash* que este traz. Estas mesmas portas série possibilitaram uma comunicação com o PC para outras finalidades para além da programação, e deste modo permitiram realizar um *debug* do funcionamento interno do Arduino, enquanto se desenvolvia a presente dissertação.

3.4 Sistema a desenvolver

Definido o *hardware* a implementar, é possível esboçar o diagrama de blocos referente ao *hardware* a implementar que explicita os principais processos realizados pela UC e pelo Computador de Bordo e quais as funções de outros equipamentos por exemplo baterias, sensores e cartão SD.

Na figura 3.5 é ilustrado o diagrama de blocos de todo o *hardware* implementado. Vários aspetos referidos no mesmo diagrama serão abordados nos capítulos seguintes, o Controlo de motor e instrumentação no capítulo 4, comunicação CAN no capítulo 5, registo de balanços energéticos em cartão SD no capítulo 6 e a *Interface* no capítulo 7. Com balanço energético entende-se consumo e regeneração.

⁴Neste caso *Universal Serial Bus* (USB)

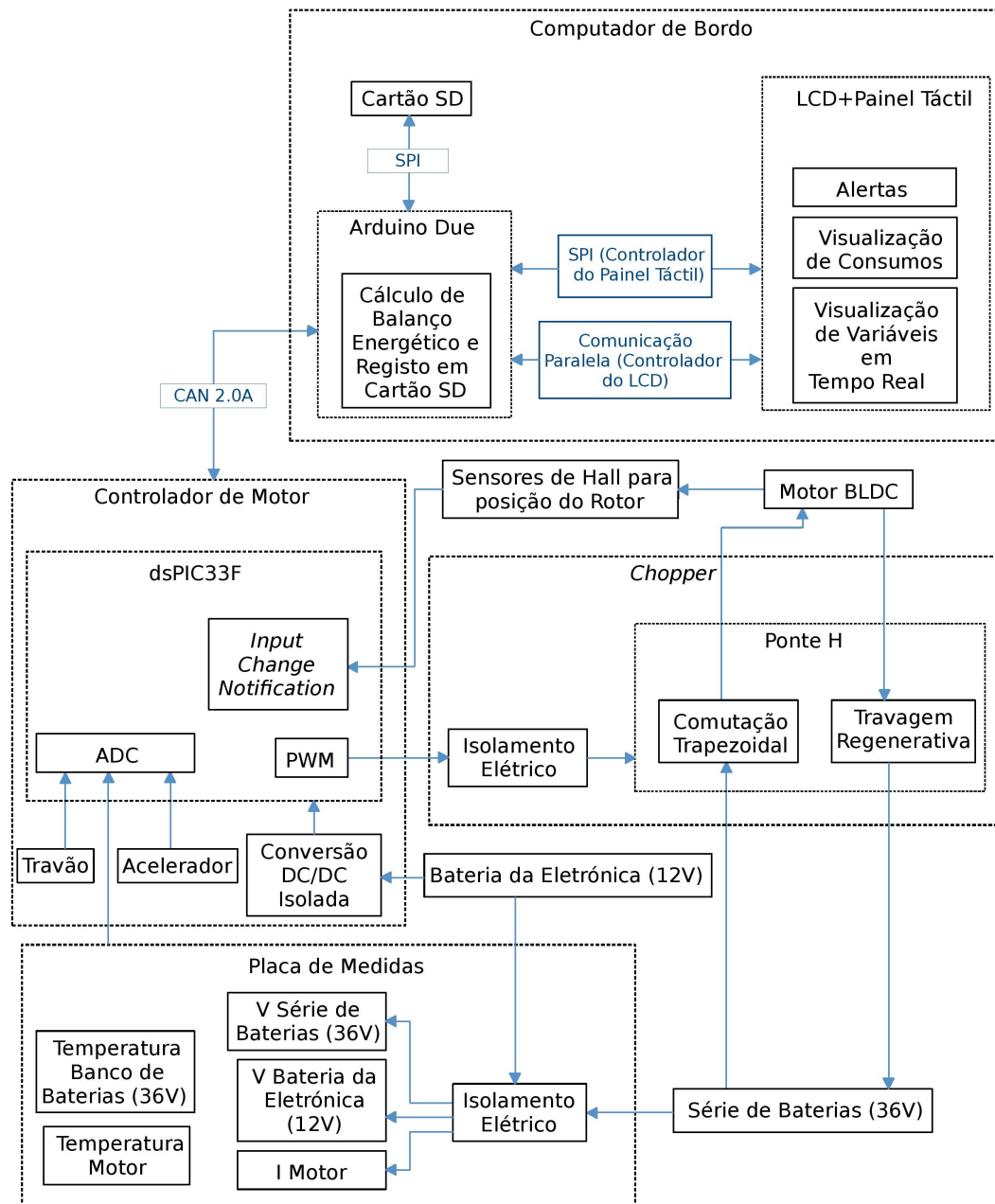


Figura 3.5: Diagrama de blocos do *hardware*.

Capítulo 4

Unidade de Controlo de motor existente e ajustes

Este capítulo serve para descrever o Controlador reaproveitado, descrevendo de forma breve o seu funcionamento. Será ainda abordada a instrumentação associada a este Controlador e as melhorias e expansões aplicadas a esta. A UC encontra-se numa placa que tem ligações elétricas com o Computador de Bordo e placa de medidas. A placa de medidas contem maioria dos sensores implementados em todo sistema de controlo. A instrumentação é abordada mais detalhadamente na secção 4.5.

4.1 Motor e alimentação da eletrónica

O motor BLDC utilizado possui 12 pares de polos no estator, 3 fases, 4 pares de polos no rotor e uma potência nominal de 500 W. A alimentação deste motor é realizada a partir de uma série de baterias de 36 V, constituída por três baterias de 12 V com o modelo LC-R127R2P1 da Panasonic. Como já referido na introdução, foi desenvolvido um sistema de travagem regenerativa em que o motor absorve energia mecânica convertendo-a em energia elétrica para assim a entregar à série de baterias.

Esta UC é eletricamente alimentada a partir de uma bateria de 12 V (do mesmo modelo) e para a obter isolamento elétrico entre os circuitos potência e eletrónica de controlo recorre-se ao conversor DC/DC isolado TEL 5 – 1211 com saída a 5 V, cuja implementação verifica-se na figura 4.1. Notar que esta mesma alimentação é usada para o Computador de Bordo.

Esta última bateria de 12 V pertence ao mesmo banco de baterias que as baterias da série de 36 V, mas não faz parte desta série. É de salientar que em um veículo elétrico esta bateria é normalmente diferente da usada para tração, para ser trocada com facilidade. A utilização de estas 4 baterias foi esquematizada na figura 3.5.

4.2 Controlo do motor BLDC

Como já mencionado na secção 3.1, o MCU do Controlador do motor tem ser substituído, o que significa que a placa de controlo tem que ser reconstruída, notar que é necessário acrescentar componentes nesta placa visto que se pretende estabelecer uma comunicação CAN. O circuito implementado do Controlador do motor BLDC encontra-se na figura 4.1, que foi implementada em uma placa perfurada. Devido à grande quantidade de sinais denominados, as entradas e saídas da UC (não do seu MCU em particular), são destacadas a azul.

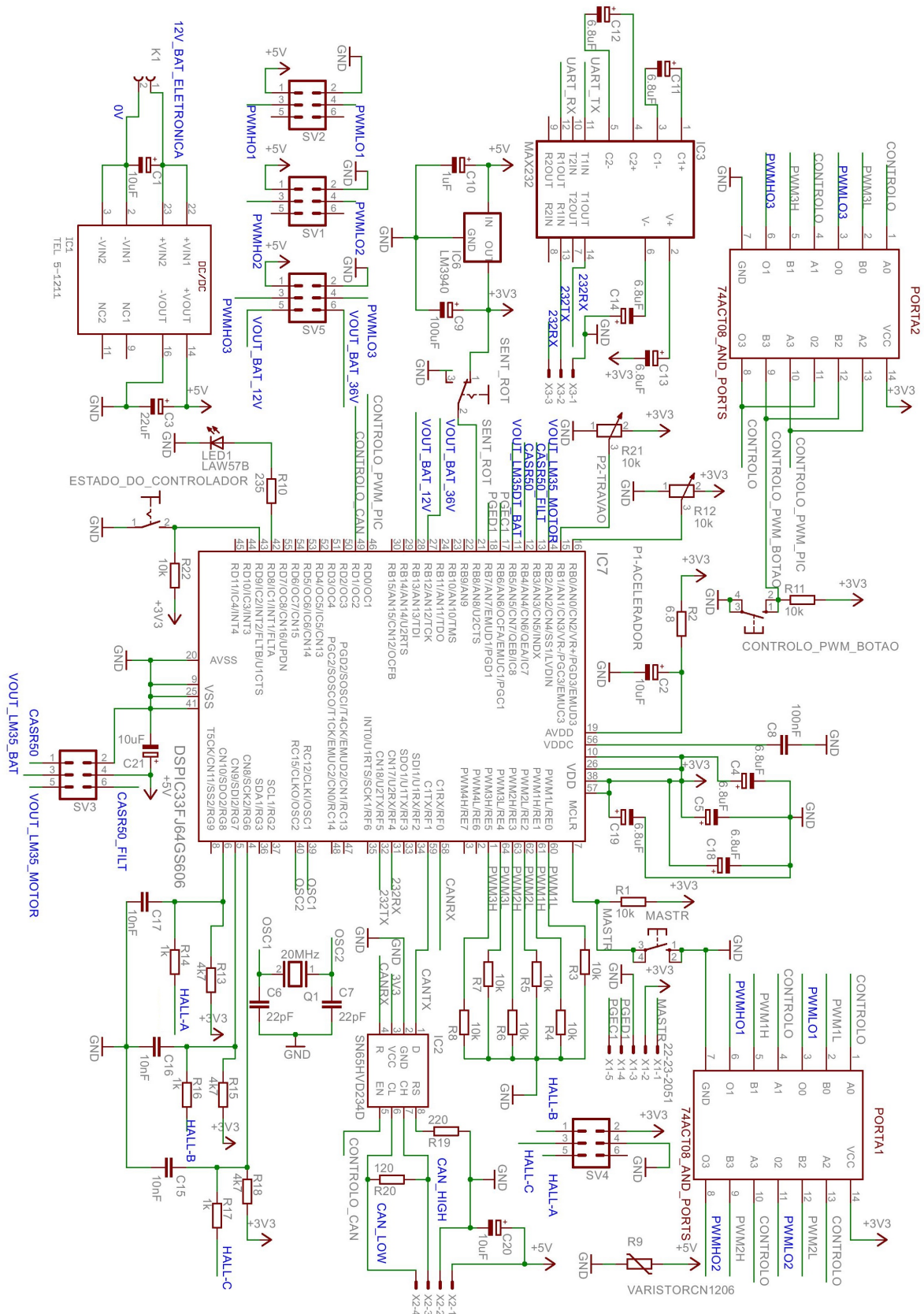


Figura 4.1: Esquema elétrico do Controlador de BLDC.

Como explicado mais adiante este controlador tem a capacidade de medir a corrente elétrica que flui de e para o motor, e caso esta corrente ultrapasse os 30 A o controlo do motor é desativado.

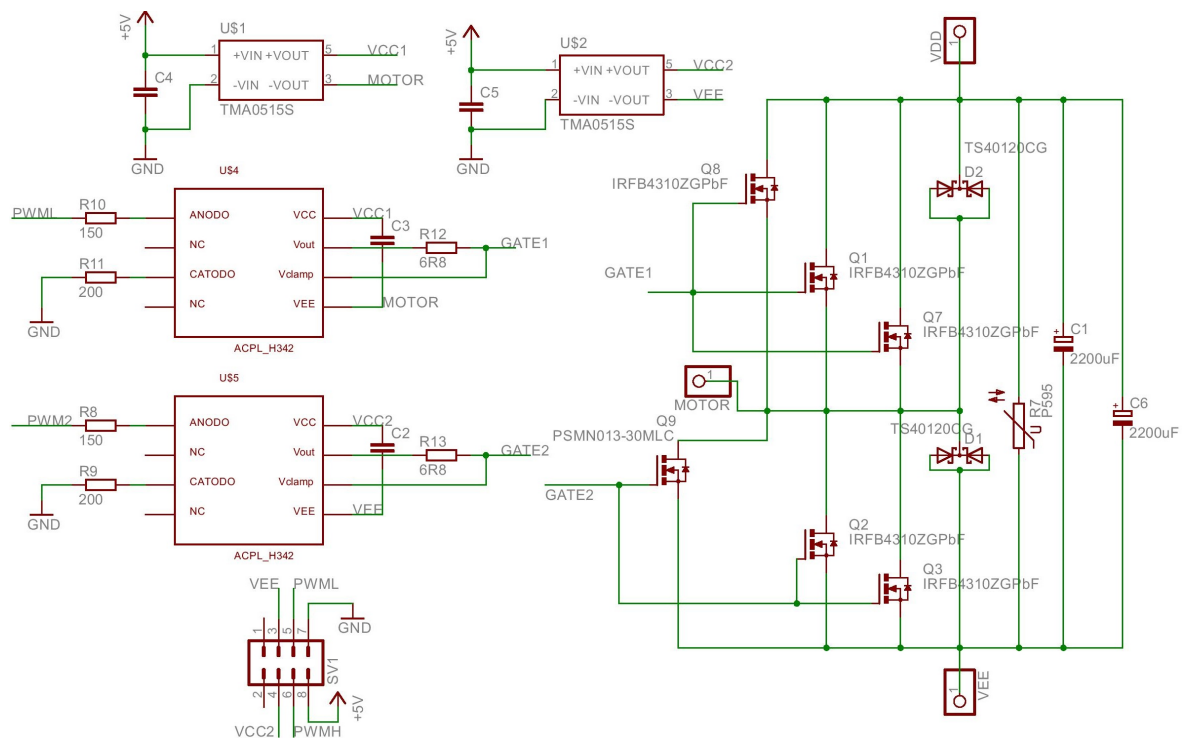


Figura 4.2: Circuito elétrico de um braço da ponte H utilizada.

Para se conseguir realizar um *debug* na UC, colocou-se na placa deste Controlador um *transceiver* RS232, o MAX232, para assim possibilitar a comunicação com um PC. Deste modo durante o desenvolvimento deste Controlador foi possível visualizar em um PC por exemplo, valores da ADC certificando que este periférico está funcional assim como ver dados recebidos pela comunicação CAN.

4.4 Interruptores e sinais de controlo

O interruptor SENT_ROT serve para alternar o sentido de rotação, o interruptor ESTADO_DO_CONTROLADOR serve para desligar ou ligar o sinal de PWM e a comunicação com o Computador de Bordo. Os sinais CONTROLO_PWM_PIC e CONTROLO_CAN dependem do interruptor ESTADO_DO_CONTROLADOR.

Por questões de segurança o sinal de PWM é externamente combinado com o sinal de controlo do MCU CONTROLO_PWM_PIC e com o sinal do botão CONTROLO_PWM_BOTAO. Este último botão permite que se desligue o sinal de PWM sem depender do controlo interno do MCU.

4.5 Instrumentação

Para além da placa de controlo foi reconstruída a placa de medidas, devido à intenção de acrescentar novos sensores ao Controlador do motor e aplicar alterações à implementação de alguns sensores. Esta placa foi referida no diagrama de blocos de todo o sistema da figura 3.5 e o circuito desta placa demonstra-se na figura 4.3. Este mesmo circuito foi implementado em uma placa perfurada.

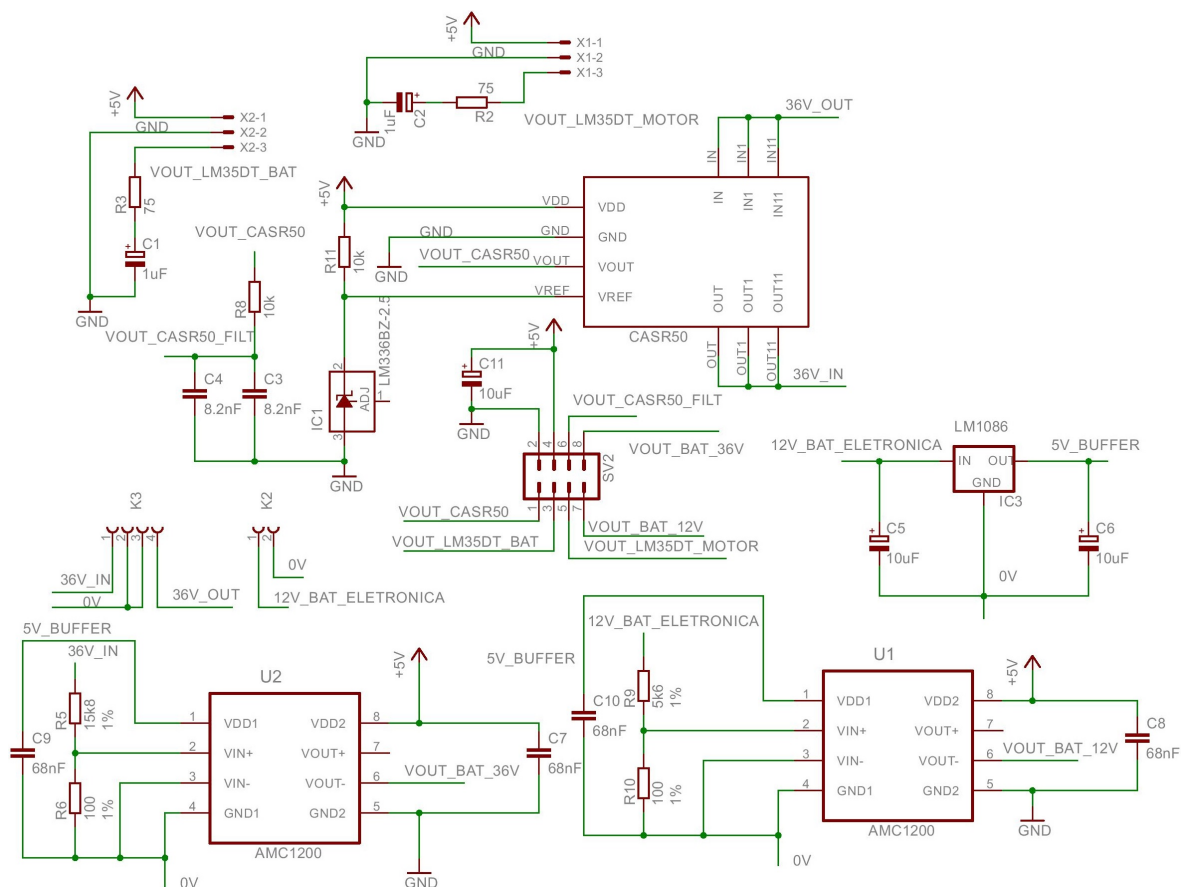


Figura 4.3: Circuito da placa de medidas.

Os sensores referentes ao motor e ao Controlador implementados na dissertação anterior são os seguintes:

- Dois potenciômetros que servem de acelerador e travão, localizados na placa de Controlo do motor. Estes controlam os *duty cycle* dos sinais PWM de comutação trapezoidal e de travagem regenerativa, respetivamente;
- 3 sensores de *Hall* para ler a posição do rotor do motor;
- Sensor de corrente CASR50, para medir a corrente que flui de e para a ponte H, ou seja, medir a corrente que flui de e para a série de baterias (36 V).

Com o trabalho da presente dissertação acrescentam-se os seguintes sensores:

- Dois sensores de temperatura com saída analógica LM35DT, aplicados na caixa do motor e no banco de baterias;
- Dois amplificadores diferenciais isolados AMC1200, para amostrar tensões na série baterias de 36 V e na bateria de 12 V sendo esta última a que alimenta eletrónica de Controlo e o Computador de Bordo.

4.5.1 Sensores de *Hall* para posição

Com estes sensores a UC tem a capacidade de medir a atual posição do rotor e consequentemente calcular a velocidade de rotação deste. A implementação destes sensores a nível de *hardware* foi completamente retirada da dissertação anterior, enquanto que a nível de *software* sofreu um simples processo de calibração para o motor BLDC utilizado, que tem 4 pares de polos no rotor. Estes sensores correspondem a entradas digitais e a comutação de cada uma destas entradas gera no MCU uma interrupção associada ao periférico *Input Change Notification*. Para calcular a velocidade de rotação é necessário saber qual frequência a que esta função é chamada, e deste modo recorre-se ao *Timer* 1, para saber o valor dos intervalos tempo entre ocorrências de interrupções externas. A ligação ao MCU deste sensores pode ser constatada no circuito da figura 4.1.

4.5.2 Sensor de corrente CASR50

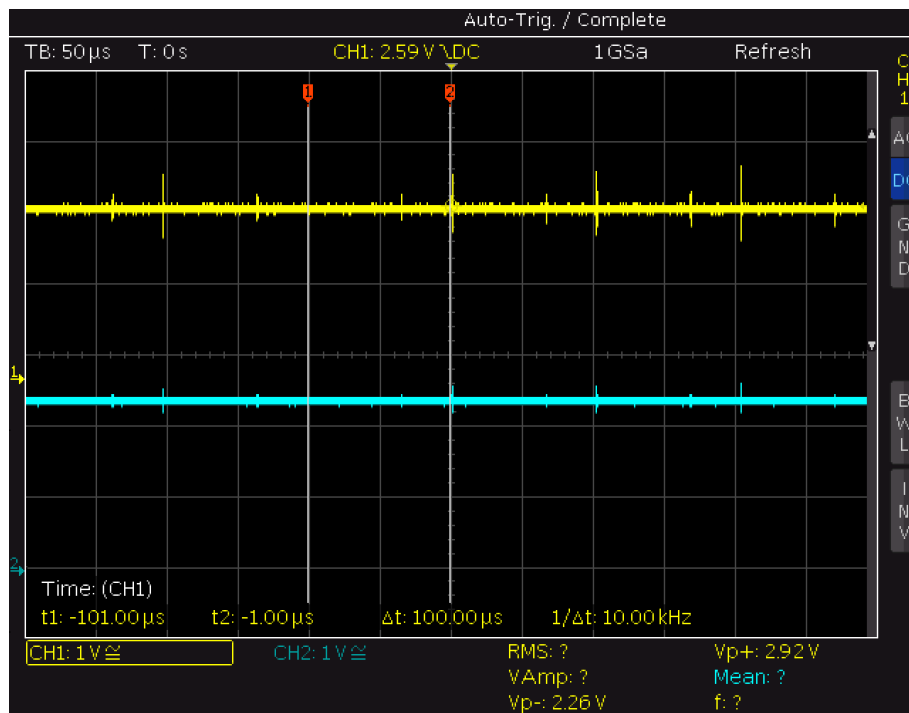
Componente reutilizado da dissertação anterior. Segundo o fabricante do sensor, as principais características são as seguintes [41]:

- Possui isolamento galvânico entre circuito primário e secundário;
- O circuito primário é percorrido pela corrente elétrica a medir enquanto que o circuito secundário tem uma saída em tensão que depende da corrente elétrica que flui no circuito primário;

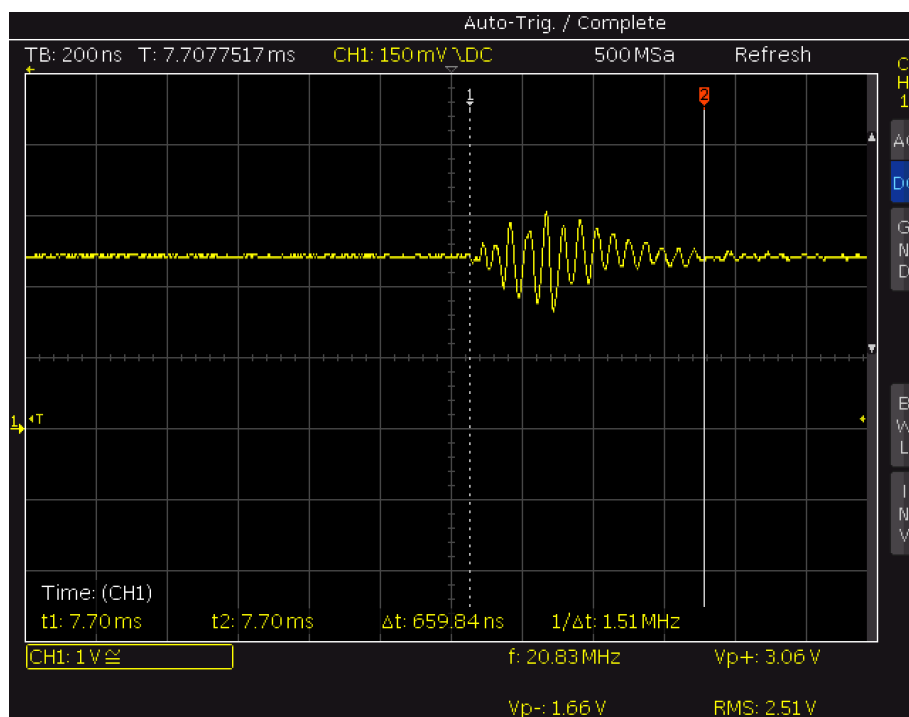
- No circuito secundário é possível aplicar uma tensão de referência de modo a centrar a saída em torno desse valor, o que é útil quando se pretende medir correntes que fluem em ambos sentidos;
- A sensibilidade é de $12,5\text{ mV/A}$.

Visto a corrente na ponte H fluir em ambos sentidos aplicou-se um tensão de referência de $2,5\text{ V}$. Este componente localiza-se na placa de medidas como se verifica no circuito da figura 4.3. No notar que a implementação do CASR50 foi realizada de maneira a que uma corrente de consumo cause um decréscimo no sinal de saída deste sensor, conseguindo assim uma maior excursão de sinal para o consumo visto serem esperadas maiores correntes de consumo do que regeneração. Assim a excursão de sinal para consumo vai de $2,5\text{ V}$ a 0 V e de $2,5\text{ V}$ a $3,3\text{ V}$ para regeneração.

Para se verificar o comportamento do circuito nas circunstâncias a que este funciona, realizaram-se medições com o sensor a ser percorrido pela corrente de consumo do motor BLDC. A figura 4.4 contém sinais de saída adquiridos no sensor de corrente utilizado num instante com consumo de 4 A .



(a)



(b)

Figura 4.4: Sinal de saída do CASR50 em duas diferentes escalas temporais.

Pela captura da figura 4.4a verifica-se a diferentes picos de tensão que se repetem a uma frequência de 10 kHz , que é a frequência de comutação da ponte H. O código fornecido do Controlador tem o periférico ADC configurado com um tempo total de leitura e conversão

de 715 ns , valor este próximo à da duração das interferências que se pretendem suprimir, duração esta por volta dos 700 ns como demonstrado na aquisição da figura 4.4b.

A leitura deste sensor foi alterada de forma a suprimir o efeito dos picos anteriores na aquisição do valor da corrente. Cada vez que o dsPIC corre função para adquirir o valor da corrente, este realiza oito conversões analógico digital seguidas e remove as duas medições de mais alto valor assim como a duas de menor valor e por fim calcula a média de com os quatro valores restantes. Com esta técnica de filtragem elimina-se as aquisições que possam ter sido afetadas com as variações repentinas de tensão (as interferências) causadas pela comutação da ponte H. Para além desta filtragem por *software*, implementou-se um filtro passa-baixo passivo RC, com uma frequência de corte de 1 kHz para tornar o sinal de uma forma geral menos ruidoso e não simplesmente por causa das interferências referidas anteriormente. Este filtro pode ser verificado na figura 4.3 e o respetivo sinal de saída no canal 2 da aquisição da figura 4.4a.

Internamente o dsPIC limita a corrente elétrica a 30 A desligando o PWM enquanto a corrente se mantiver acima deste valor.

4.5.3 Sensores de temperatura LM35DT

Estes sensores possuem uma sensibilidade de $10\text{ mV}/^{\circ}\text{C}$. A justificação de ter escolhido este sensor passa pelos seguintes motivos:

- Calibração por defeito em graus *Celsius*;
- Intervalo de temperaturas admissíveis apropriado para a aplicação, sendo este intervalo de -55°C a 150°C ;
- Alta linearidade permite fácil leitura. Não linearidades tipicamente de $0,25^{\circ}\text{C}$;
- Encapsulamento volumoso (TO -220), que facilita experiências laboratoriais e montagem em equipamento a que estes sensores se destinam (motor e banco de baterias).

A implementação destes sensores constata-se na 4.3, pode-se verificar que se introduziram amortecedores RC, tal como sugerido na ficha de dados do componente [42], para melhorar a tolerância à capacidade dos condutores.

4.5.4 Amplificadores isolados AMC1200

Para conseguir uma amostragem da tensão da baterias mantendo isolamento elétrico entre potência e eletrónica de controlo recorreu-se aos amplificadores com isolamento galvânico AMC1200. Estes sensores têm a capacidades de amplificar tensões diferenciais com um ganho diferencial internamente definido de 8. A sua entrada diferencial suporta no máximo 250 mV independentemente da polaridade e possui um resistência interna de $28\text{ k}\Omega$. Para uma fácil

consulta da ficha de dados do AMC1200, colocou-se no anexo A um excerto desse documento contendo os aspetos mais relevantes na aplicação deste componente no presente trabalho [43].

No sentido de adquirir um circuito de amostragem que respeite as condições anteriores, realizou-se dois divisores resistivos (um para cada tensão a amostrar) cujo circuito equivalente apresenta-se na figura 4.5.

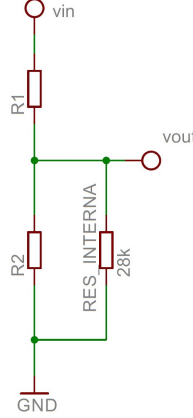


Figura 4.5: Divisor resistivo equivalente para acondicionamento das tensões das baterias.

Para ambos os sensores arbitrou-se uma corrente nos divisores resistivos de 2 mA caso as baterias tenham as suas tensões nominais. Segundo o fabricante das baterias utilizadas estas devem ser carregadas no máximo com $14,5\text{ V}$ [44] e para efeitos de dimensionamento dos circuitos de acondicionamento, admitiu-se que esta é a tensão máxima em cada bateria. Deste modo a tensão máxima na série de baterias é de $43,5\text{ V}$. Para dimensionamento dos divisores resistivos, assumiu-se que as entradas dos amplificadores tenham o valor de tensão máximo recomendado de 250 mV caso a tensão nas baterias seja máxima. Cumprindo as especificações anteriores determinou-se o sistemas de equações da expressão 4.1.

$$\begin{cases} v_{out\ max} = V_{in\ max} * \frac{R_{in} * R_2}{R_{in} + R_2} / \left(R_1 + \frac{R_{in} * R_2}{R_{in} + R_2} \right) \\ I_N = V_N / \left(R_1 + \frac{R_{in} * R_2}{R_{in} + R_2} \right) \end{cases} \Leftrightarrow \begin{cases} v_{out\ max} = V_{in\ max} / \left(1 + \left(R_1 / \frac{R_{in} * R_2}{R_{in} + R_2} \right) \right) \\ I_N = V_N / \left(R_1 + \frac{R_{in} * R_2}{R_{in} + R_2} \right) \end{cases} \quad (4.1)$$

Em que:

- $v_{out\ max}$ é a tensão máxima de saída dos divisores resistivos, ou seja na entrada dos amplificadores, com o valor de 250 mV ;
- R_{in} é a resistência interna das entradas de sinal dos amplificadores, como já referido de $28\text{ k}\Omega$;

- $V_{in\ max}$ é a tensão de entrada máxima, ou seja, a tensão máxima nas baterias com os valores $43,5\ V$ para a série (alimentação do motor) e $14,5\ V$ para a bateria de alimentação da eletrônica;
- V_N é a tensão nominal das baterias, $36\ V$ para o série e $12\ V$ para a bateria de alimentação da eletrônica;
- I_N é a corrente do divisor resistivo caso as baterias tenham a tensão nominal, esta foi arbitrada com o valor de $2\ mA$ para ambos amplificadores.

Aplicando para os valores anteriores nas equações da expressão 4.1, conclui-se que para o circuito de acondicionamento da tensão da série de baterias $R_1 \approx 17,89\ k\Omega$, $R_2 \approx 103,83\ \Omega$ e para a circuito de acondicionamento da tensão da bateria única $R_1 \approx 5,89\ k\Omega$ e $R_2 \approx 103,83\ \Omega$.

Estando o circuito de acondicionamento dimensionado é possível adquirir uma função de transferência, que relacione a tensão das baterias a medir com o sinal $v_{in\ ADC}$ a converter pelo dsPIC33F. Esta função é deduzida na expressão 4.2.

$$v_{in\ ADC} = v_{out} * G = 2,55 \pm \left(V_{in} / \left(1 + \left(R_1 / \frac{R_{in} * R_2}{R_{in} + R_2} \right) \right) \right) * G \Leftrightarrow \quad (4.2)$$

$$V_{in} = (2,55 \pm v_{in\ ADC}) * \left(1 + \left(R_1 / \frac{R_{in} * R_2}{R_{in} + R_2} \right) \right) / G$$

G é o ganho interno do amplificador. Este componente possui uma saída inversora e outra não inversora o que justifica o sinal de soma ou subtração da expressão anterior. As saídas são centradas em $2,55\ V$.

A expressão anterior tem utilidade para o processamento das entradas analógicas em que são colocadas os sinais de saída dos amplificadores do AMC1200 e as saídas utilizadas são as inversoras, como se verifica no esquema elétrico da figura 4.3. O uso das saídas inversoras em ambos amplificadores justifica-se pelo facto de o MCU da UC conseguir amostrar e processar tensões até $3,3\ V$ e deste modo conseguir uma maior excursão de sinal (de $2,55\ V$ a $0\ V$). As resistências dos divisores resistivos implementados têm precisão de $1\ \%$.

Notar que o valor do ganho referido anteriormente aplica-se para uma saída diferencial, visto nesta aplicação apenas se usar uma das saídas de cada sensor o ganho em questão é metade do diferencial e assim o ganho é 4.

No circuito do Controlador do motor da figura 4.1, notam-se estas entradas analógicas do dsPIC33F.

A implementação destes circuitos requereu adquirir uma tensão de $5\ V$ para alimentação dos amplificadores com a mesma referência que a tensão da série de baterias e bateria de alimentação da eletrônica (notar que a bateria de alimentação de eletrônica e a série de alimentação do motor têm a mesma referência). Para conseguir esta tensão de $5\ V$ utilizou-se um regulador de tensão linear LM1086 e a sua implementação pode ser verificada na mesma figura.

Capítulo 5

Desenvolvimento da comunicação CAN

Neste capítulo será abordada a comunicação implementada entre UC do motor BLDC e o Computador de Bordo responsável pelo cálculo, registos energéticos e *Interface* com o utilizador. Será mencionado qual o *transceiver* utilizado, para que se realize uma *interface* entre controladores CAN e barramento. Neste capítulo são descritos quais os parâmetros/variáveis que circulam no barramento. É explícito o trabalho desenvolvido acerca a comunicação em cada unidade, e deste modo mencionadas quais as funções de cada unidade na rede CAN.

5.1 *Transceiver* CAN

O *transceiver* utilizado foi o SN65HVD234D da Texas Instruments. A escolha deste modelo deveu-se ao facto de este ser destinado a MCUs que funcionam a 3,3 V e possuir uma entrada que quando desativada o *transceiver* entra num modo de baixo consumo. Para seguir as recomendações da ISO 11898, colocou-se nos limites físicos do barramento resistências de $120\ \Omega$ [45], assim como demonstrado nas figuras 4.1 e 5.5. O *transceiver* utilizado para a UC incluí-se na mesma placa enquanto que o *transceiver* CAN do Computador de Bordo inclui-se em uma placa perfurada separada.

5.2 Identificação de parâmetros e configurações do protocolo

Na tabela 5.1 registam-se as variáveis/parâmetros transmitidas entre ambas unidades, mencionando o tipo de variável que estes parâmetros correspondem na programação e respetivo comprimento.

Tabela 5.1: Parâmetros transmitidos pela rede CAN.

| Parâmetro/Variável | Tipo de variável | Comprimento [bytes] | ID em numeração hexadecimal |
|---|-------------------|------------------------|-----------------------------------|
| Estado do Controlador, Ligado ou Desligado | Inteiro sem sinal | 1 | 0 |
| Posição do Acelerador [%] | Inteiro sem sinal | 1 | 1 |
| Posição do Travão [%] | Inteiro sem sinal | 1 | 2 |
| Sentido de Rotação, Ligado ou Desligado | Inteiro sem sinal | 1 | 3 |
| Velocidade de Rotação [RPM] | Inteiro sem sinal | 2 | 4 |
| Corrente no Motor [A] | <i>Float</i> | 4 | 5 |
| Corrente no Conversor Buck [A] | <i>Float</i> | 4 | 6 |
| Limite de Corrente no Motor (constante) [A] | <i>Float</i> | 4 | 7 |
| Tensão na série de Baterias [V] Valor Nominal de 36 V | <i>Float</i> | 4 | 8 |
| Tensão na Bateria de Alimentação de Eletrónica [V] Valor Nominal de 12 V | <i>Float</i> | 4 | 9 |
| Temperatura no Motor [V] | <i>Float</i> | 4 | A |
| Temperatura no Banco de Baterias [°C] | <i>Float</i> | 4 | B |
| Limite de Temperatura (constante) [°C] | <i>Float</i> | 4 | C |

O parâmetro “Sentido de Rotação” depende de um interruptor para controlar o sentido de rotação, que se pode constatar no circuito da figura 4.1. O parâmetro “Limite de Temperatura” indica a temperatura máxima em graus *Celsius* admissível nas baterias e motor.

O parâmetro “Corrente no Conversor *Buck*” corresponde à corrente de um conversor *Buck* que converte a tensão da série de baterias para o valor da bateria de alimentação da eletrónica (12 V) para assim assegurar a carga desta. Este parâmetro é transmitido com o valor de 0 A visto não se ter construído um conversor DC/DC com esta função.

A taxa de transmissão é de 250 kb/s, o formato adotado das tramas é o 2.0A ou *standard* com identificadores de 11 bits visto não haver a necessidade de identificadores de 29 bits. O arbítrio dos identificadores sucedeu-se tendo em conta o tipo de variável em questão, e dentro do mesmo tipo de variável os IDs foram atribuídos conforme a importância de cada variável.

A figura 5.1 contém uma captura de osciloscópio de uma mensagem CAN transmitida no sistema de comunicação elaborado na presente dissertação.

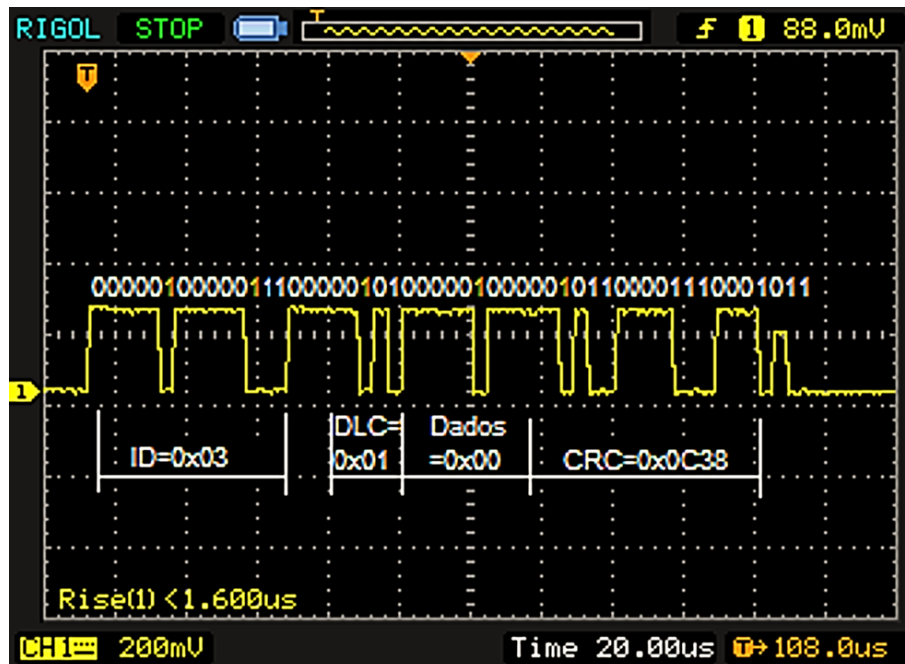


Figura 5.1: Captura de trama CAN 2.0A.

A captura anterior realizou-se com o terminal de referência (ou massa) ligada no linha CAN *Low* do barramento CAN. Verifica-se pelo ID que a variável transmitida foi o sentido de rotação, com a dimensão de 1 *byte*, nesta mesma figura destacaram-se os *bits* resultantes da inserção de *bits*, usada no protocolo CAN. Pela mesma figura confirma-se a taxa de transmissão de 250 *kbits/s*.

5.3 dsPIC33FJ64GS606 - Unidade de Controlo

De seguida é demonstrado o trabalho desenvolvido na UC no que toca à comunicação CAN e é explicado quais as funções desta na rede CAN.

5.3.1 Biblioteca CAN

O código/biblioteca CAN usado no dsPIC33FJ64GS606 foi adaptado de uma biblioteca destinada a um MCU da família dsPIC30F, biblioteca esta desenvolvida na dissertação apresentada em 2010 com o título “Implementação e testes de robustez do protocolo tempo-real FTT-CAN” [46]. Esta biblioteca foi ligeiramente modificada para compatibilidade com o novo MCU, ainda foi acrescentada a possibilidade de enviar tramas remotas¹ e distinguir estas das de dados caso sejam recebidas.

¹Ou RTR

Esta biblioteca, para sua utilização fornece uma estrutura de dados que corresponde a uma mensagem CAN, cujos membros são dados, comprimento do campo de dados, ID, e tipo de mensagem (dados ou remota). O membro de dados consiste em um vetor em que cada elemento tem um *byte* de dimensão, visto o campo de dados de uma mensagem CAN variar de um em um *byte*.

Recorre aos periféricos *Enhanced CAN* (ECAN) e *Direct Memory Access* (DMA), em que o DMA permite a cópia entre registos de periféricos, os *Special Function Registers* (SFRs), e variáveis localizadas na *Random Access Memory* (RAM) com um esforço do CPU minimizado. Neste caso os registos de periféricos contêm dados recebidos e a enviar pelo ECAN.

5.3.2 *Hardware* relacionado com a comunicação

Para além do *transceiver* referido na secção 5.1, acrescentou-se o interruptor designado “Estado do Controlador” que tem a si associado a um parâmetro CAN com o mesmo nome. Este interruptor pode ser verificado no circuito da placa de controlo da figura 4.1. Por intermédio do dsPIC33F este interruptor desabilita o *enable* induzindo o *transceiver* ao modo *Sleep*.

A implementação da comunicação CAN induziu uma alteração da UC, que consistiu do aumento da frequência de oscilação de 7,37 *MHz* para 20 *MHz* recorrendo a um oscilador externo que pode ser verificado na figura 4.1. A justificação da medida anterior foi conseguir taxas de comunicação mais elevadas e com valores mais exatos.

5.3.3 Envio, receção e processamento de mensagens CAN

O enquadramento, ou seja circunstâncias, em que são enviadas para o barramento mensagens CAN por parte da UC é descrito no fluxograma da figura 5.2.

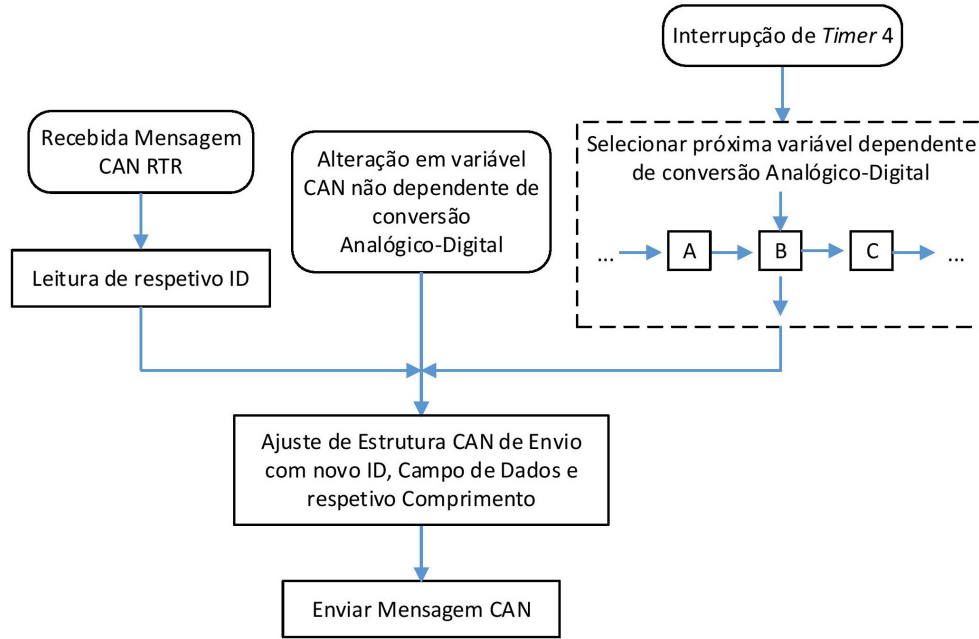


Figura 5.2: Fluxograma transmissão de mensagens CAN no dsPIC33F.

Todas as variáveis resultantes de conversões analógico-digital (potenciômetros de acelerador e travão e variáveis *float*), são transmitidas periodicamente e sequencialmente visto não ser recomendado serem enviadas sempre que se verifica uma alteração. Este método justifica-se pelo facto dos valores destas variáveis serem adquiridos por intermédio de conversões analógico-digital de sinais gerados por sensores que apresentam nas suas saídas ruídos significativos, causando assim alterações frequentes e pouco significantes em torno do valor correto. Para enviar estas variáveis alternadamente, configurou-se o *Timer 4* a gerar interrupções de 100 em 100 *ms*, e nesta rotina é enviada uma das variáveis em questão, alternando de variável cada vez que a rotina é executada. Por este método evita-se a ocupação quase constante e desnecessária do barramento. Notar que desta sequência de envio excluem-se os parâmetros “Limite de Corrente no Motor” e “Limite de Temperatura” visto serem constantes.

Apesar de o envio de certas mensagens seja distribuído por intervalos de tempo controlados, esta comunicação não se trata de uma comunicação *Time-Triggered* visto não serem transmitidas mensagens de referência.

Para realizar a correspondência entre parâmetro/variável e ID CAN, todas as variáveis/-parâmetros CAN tratam-se de vetores de três posições. O conteúdo deste vetores é explicito na expressão 5.1.

$$Variável[3] = \{Valor\ Atual, \acute{U}ltimo\ Valor\ Enviado, ID\ CAN\} \quad (5.1)$$

À exceção das variáveis dependentes de conversões analógico-digital, sempre a que uma variável CAN é atribuído um valor, é executada uma função em que se comparam as duas primeiras posições do vetor. Caso estas tenham valores diferentes a estrutura de mensagem CAN é ajustada com o um novo ID, comprimento do campo de dados e dados (valor do

campo de dados), e então é enviado uma nova mensagem CAN. Qualquer uma das variáveis é enviada caso seja recebido uma trama remota com o respetivo ID.

Como referido na subsecção 5.3.1 os dados a enviar são colocados num vetor cuja dimensão de cada elemento é de um *byte*. Certos parâmetros CAN correspondem a variáveis que ocupam mais que um *byte* e têm que ser colocados no vetor de envio. Para atingir esta última finalidade recorreu-se a operações de deslocamento de *bits* (ou *bit shifts*) e de lógica de *bits*. Este processo é esquematizado no diagrama da figura 5.3.

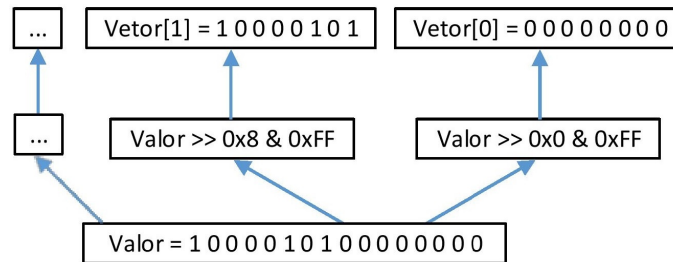


Figura 5.3: Inserção de valor de parâmetros CAN em vetor de envio.

As operações binárias referidas anteriormente, na linguagem C não se aplicam a variáveis do tipo *float*. Para solucionar a limitação anterior recorreu-se ao tipo de dados especial existente nesta linguagem de programação designada *union*². Este tipo de dados permite que o mesmo endereço, ou seja uma variável, possa ter tratado como diferentes tipos de dados. Deste modo antes de se editar a estrutura CAN referente à mensagem a enviar, recorre-se a uma união conseguindo colocar o valor binário da variável *float* em questão em uma variável do tipo *int* e assim efetuar as operações binárias necessárias.

5.4 Arduino Due - Computador de Bordo

Nesta secção é demonstrado o trabalho desenvolvido no Computador de Bordo no que toca à comunicação CAN e é explicado quais as funções desta na rede CAN.

5.4.1 Capacidades CAN do ATSAM3X8E e ligações elétricas

O MCU ATSAM3X8E possui dois controladores CAN, o CAN0 e CAN1 e o controlador usado é o CAN0. Este MCU no seu *hardware* interno contém 8 caixas de correios por cada controlador, em que cada uma pode servir de *buffer* de mensagens recebidas e enviadas. A implementação do *transceiver* CAN realizou-se numa placa externa ao Arduino, o circuito desta verifica-se na figura 5.5, e as ligações realizadas no Arduino estão ilustradas na figura 5.4.

²União

Arduino Due (apenas terminais acessíveis representados)

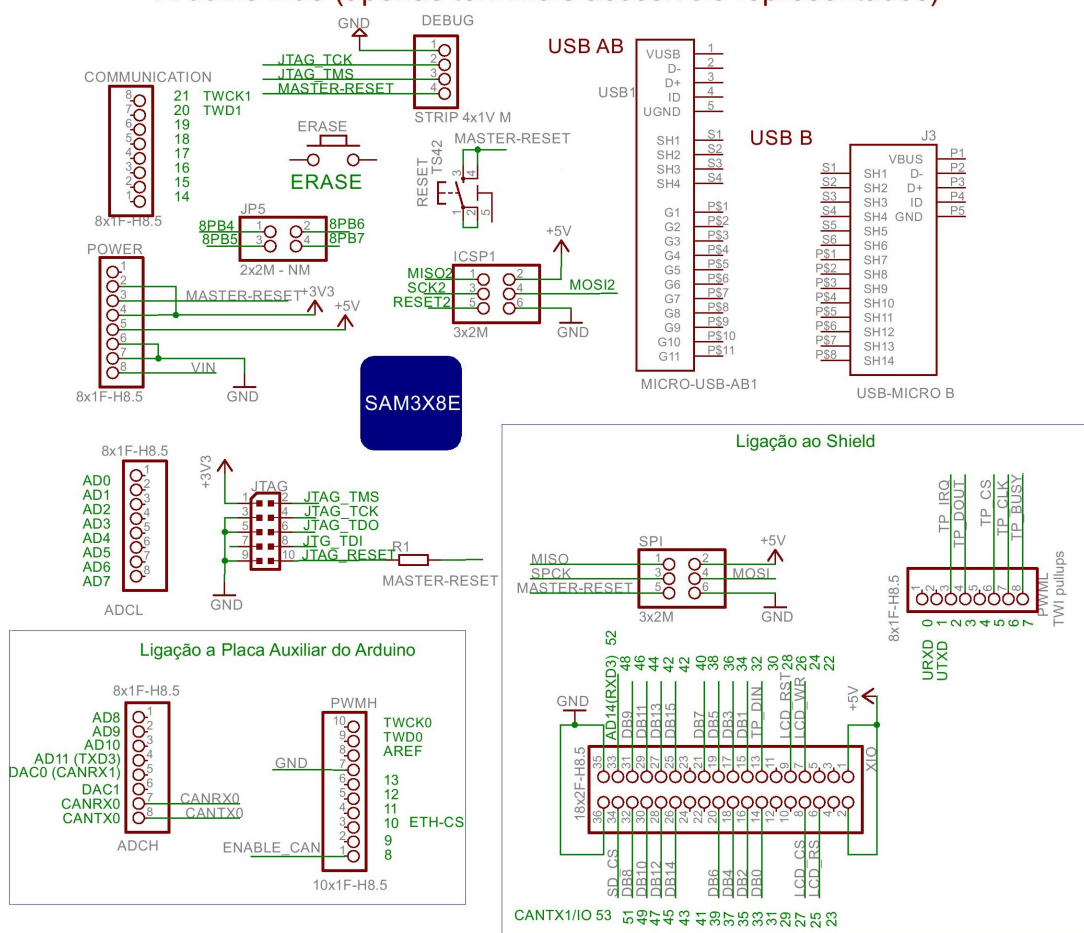


Figura 5.4: Ligações elétricas no Arduino Due.

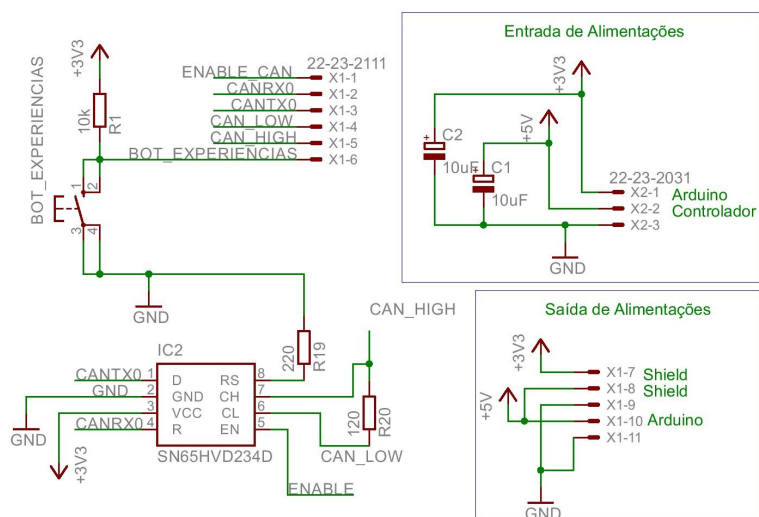


Figura 5.5: Placa do *transceiver* CAN do Computador de Bordo.

Notar que a figura 5.4 mostra apenas os terminais acessíveis da placa Arduino omitindo a eletrônica interna deste e os terminais não utilizados têm nomes/funções legendados unicamente a verde. Verificam-se ainda as ligações referentes às implementações do LCD, do painel tátil e do cartão SD, implementações estas serão abordadas nos capítulos seguintes.

5.4.2 Biblioteca CAN

A biblioteca CAN usada não é oficial, trata-se de uma biblioteca desenvolvida para a placa Arduino em questão, por uma entidade individual e publicada *online* [47]. Esta biblioteca foi incluída no IDE oficial do Arduino, usado para programar o módulo Arduino Due com o qual se desenvolveu o Computador de Bordo. Na data em que se desenvolveu não existia qualquer biblioteca oficial destinada à comunicação CAN para o módulo Arduino usado, o que justificou a pesquisa por código desenvolvido por terceiros. Esta biblioteca foi igualmente ajustada de maneira a que possibilitasse o envio de tramas remotas e distinção destas das tramas de dados na recepção.

Esta biblioteca tal como a usada no dsPIC, fornece uma estrutura de dados que corresponde a uma mensagem CAN, e a utilização desta biblioteca é orientada a objetos em que os objetos existentes são os ports CAN existentes no SAM3X8E. Por predefinição esta biblioteca configura 7 caixas de correio como *buffers* de recepção e 1 como de envio, este aspeto não foi alterado pois o Computador de Bordo apenas envia mensagens CAN durante a sua inicialização tal como explicado de seguida.

5.4.3 Envio, recepção e processamento de mensagens CAN

Sempre que se inicia o Computador de Bordo, o MCU incorporado neste, envia uma sequência de tramas remotas de forma a que adquira os valores de todas as variáveis relacionadas com o Controlador. As várias funcionalidades do Computador de Bordo, explicadas no capítulo 7, são disponibilizadas apenas após este último processo estar concluído. Naturalmente que o Computador de Bordo após o processo anterior, está disponível para receber e processar novamente os diferentes parâmetros. O ciclo de inicialização de parâmetros CAN é esquematizado no fluxograma da figura 5.6.

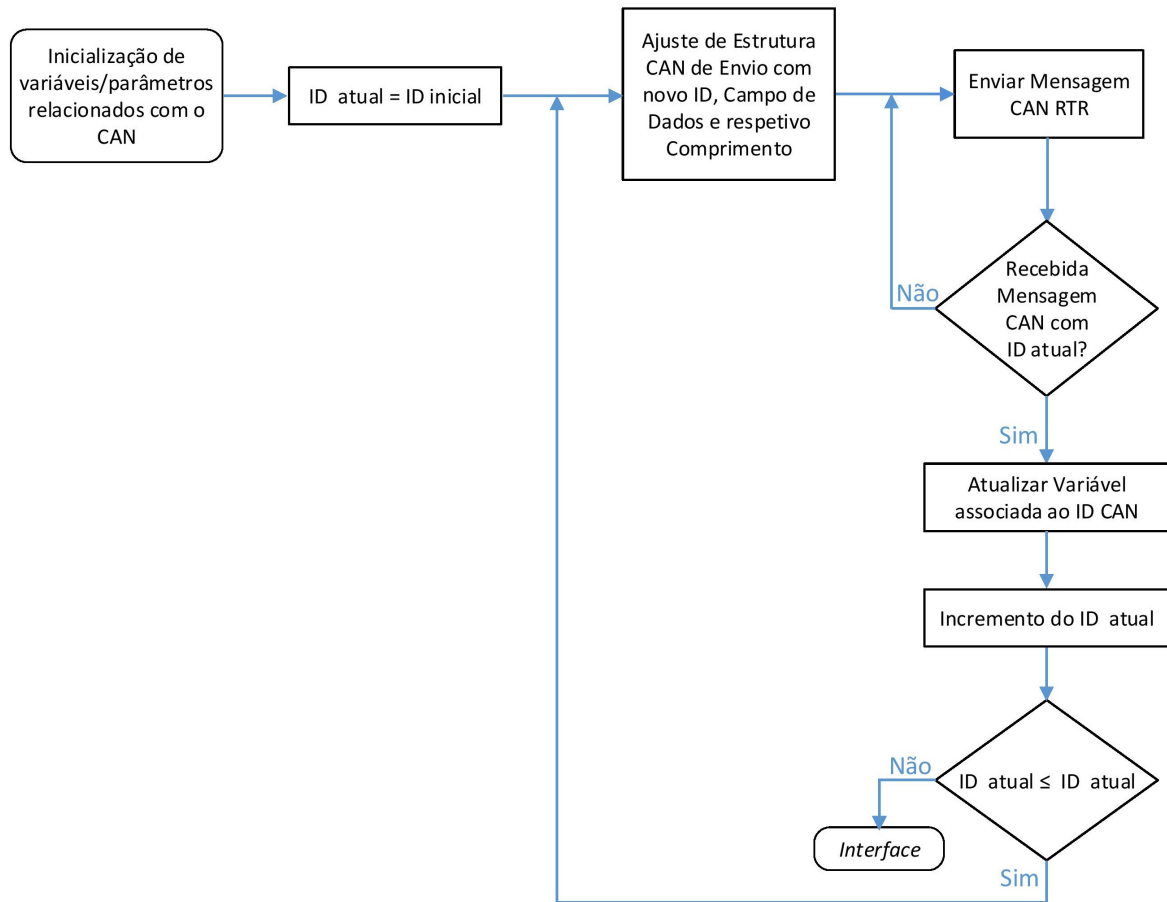


Figura 5.6: Fluxograma de inicialização de parâmetros CAN.

Capítulo 6

Computador de Bordo - Balanços energéticos e respetivo registo

Como já referido na introdução, o Computador de Bordo, tem a capacidade de calcular e registar os consumos e regeneração energéticas e o trabalho desenvolvido sobre estes últimos registos será abordado mais detalhadamente na secção 6.4.

Para uma fácil consulta pelo utilizador estes balanços energéticos são registados em ficheiros de texto colocados em um cartão SD. A comunicação com um cartão SD implicou uma pesquisa e estudo de formas de comunicação com estes componentes, com o intuito de decidir como estabelecer uma comunicação entre Arduino e cartão SD, o que resultou em toda a secção 6.2. Para se localizarem no tempo os registos energéticos recorreu-se a um *Real Time Clock*, incluído no MCU do Arduino.

Neste capítulo serão referidas oportunamente certos aspetos sobre a *Interface* que será abordada mais detalhadamente no capítulo 7.

6.1 Inicialização do Computador de Bordo

De forma a referir as funcionalidades do Computador de Bordo e qual *hardware* relacionado a este, a sua inicialização é esquematizada no fluxograma da figura 5.6. A inicialização envolve a configuração de diversos periféricos do Arduino Due e de outros equipamentos interligados a este como o cartão SD, módulo LCD e respetivo painel tátil.

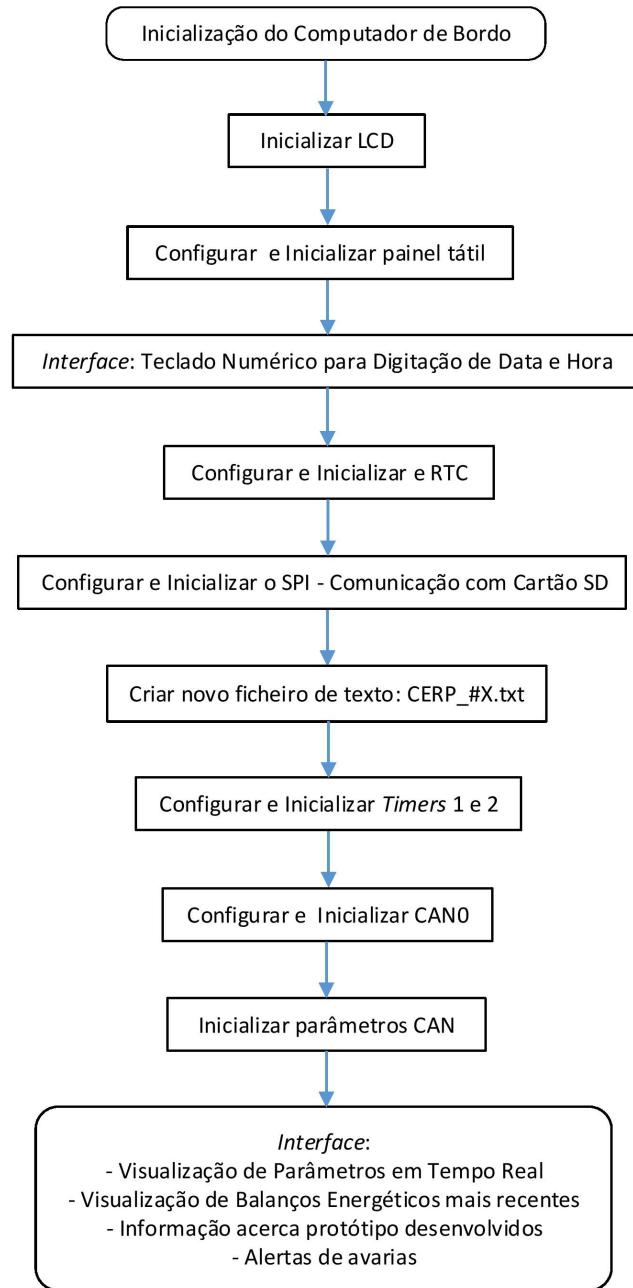


Figura 6.1: Fluxograma da inicialização do Computador de Bordo.

No fluxograma anterior refere-se o processo “Inicializar parâmetros CAN”, que já foi abordado na subsecção 5.4.3.

A implementação do *Real Time Clock* (RTC) é descrita na secção 6.3, este periférico é configurado a parte da interface como indica o processo “Interface: Teclado Numérico para Digitação de Data e Hora”, e esta componente da Interface é descrita mais adiante na subsecção 7.4.1. Verifica-se o recurso a *Timers* que são necessários para realizar os balanços energéticos, processo abordado mais detalhadamente na secção 6.4. Tal como referido no início do capítulo recorre-se a um cartão SD para registo de consumos energéticos, e o trabalho

em torno deste último componente é descrito nas secções 6.4 e 6.5.

São mencionadas as funcionalidades da *Interface* a serem abordadas no capítulo 7.

6.2 Comunicação com cartão SD

Nesta secção será feita uma breve descrição dos diferentes modos de *interface* com os cartões SD e respetivos protocolos de comunicação. No final será justificado o protocolo utilizado e ilustradas as ligações elétricas associadas.

6.2.1 *Pinout* e diferentes *interfaces* nos cartões SD

A *interface* com um Cartão SD pode ser realizada por três diferentes protocolos, que se dividem em dois modos: o modo SPI e o modo SD. A figura 6.2 mostra o diagrama de pinos de um cartão SD.

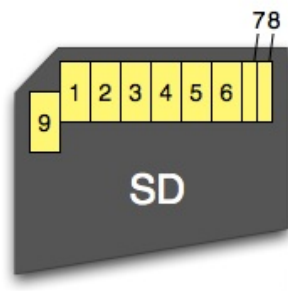


Figura 6.2: Diagrama de pinos de um cartão SD [13].

A função de cada pino conforme o modo de comunicação demonstra-se na tabela 6.1.

Tabela 6.1: Funções dos pinos de cartão SD [16].

| Pino | Acrónimo | | Função | |
|------|----------|----------|-----------------------|--|
| | Modo SD | Modo SPI | Modo SD | Modo SPI |
| 1 | DAT3 | CS | Linha de dados 3 | <i>Chip Select/Slave Select</i> (\overline{SS}) |
| 2 | CMD | DI/MOSI | Linha de comandos | <i>Master Out Slave In</i> |
| 3 | VSS1 | | Massa | |
| 4 | VDD | | Tensão de alimentação | |
| 5 | VSS2 | | Massa | |
| 6 | CLK | | <i>Clock</i> | |
| 7 | DAT0 | DO/MISO | Linha de dados 0 | <i>Master In Slave Out</i> |
| 8 | DAT1 | IRQ | Linha de dados 1 | Não usado ou <i>Interrupt Request</i> |
| 9 | DAT2 | NC | Linha de dados 2 | Não usado |

6.2.2 *Serial Peripheral Interface*

Este protocolo foi desenvolvido pela Motorola. A troca de informação segue a topologia *Master-Slave*. Cobre o nível físico e o nível de ligação de dados do modelo OSI. Neste protocolo as diferentes estações comunicam a dois fios, o *Serial Data In* (SDI) e o *Serial Data Out* (SDO). Os nomes anteriores aplicam-se no ponto de vista de cada estação e geralmente neste protocolo as linhas designam-se *Master In Slave Out* (MISO) e *Master Out Slave In* (MOSI). Para além das linhas anteriores este protocolo implica um terceiro condutor que se trata da linha de *clock*, no qual o mestre coloca um sinal com a mesma frequência que as linhas de dados para todas as estações se sincronizem conhecendo os instantes em que devem amostrar o sinal ou enviar. Portanto trata-se de um protocolo síncrono.

A figura 6.3 mostra a topologia de ligações de comunicação SPI com vários escravos.

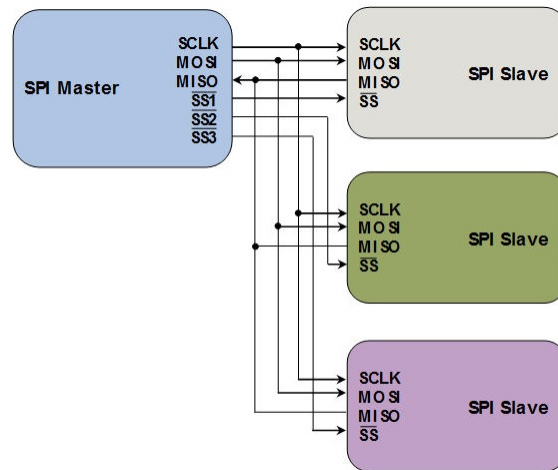
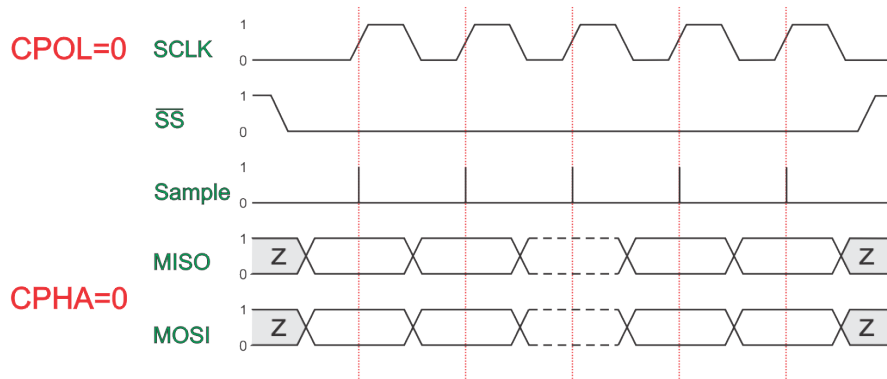


Figura 6.3: Topologia de ligações SPI [14].

Verifica-se a existência de mais um sinal, o *Slave Select* (SS). Antes de iniciar a comunicação o *Master* ativa a entrada SS do *Slave* (geralmente a 0 V) com o qual pretende comunicar.

Os dados enviados são divididos em palavras de 8 bits em que o *slave* responde sempre, mesmo no caso em que não tenha quaisquer dados a enviar e nesse caso a resposta é uma palavra com valor nulo. A figura 6.4 mostra os sinais envolvidos em uma comunicação SPI, verificam-se os aspetos anteriormente mencionados como o sinal *clock* e SS ativos durante a transmissão [14].

Figura 6.4: Sinais e tempos de amostragem de uma comunicação SPI com $CPOL = 0$ e $CPHA = 0$ [15].

No SPI os dados são transmitidos e amostrados em níveis de tensão diferentes do *clock* e é possível definir qual a polaridade da linha de *clock* e em quais transições desta se amostram os dados, sendo sempre no mesmo tipo de transições (positivas ou negativas). Para isso o SPI possui dois parâmetros o *Clock Polarity* (CPOL) e o *Clock Phase* (CPHA). Na tabela 6.2 faz-se a correspondência entre valores de ambos parâmetros e implicações e significados destes valores na comunicação.

Tabela 6.2: Valores e significados dos parâmetros CPOL e CPHA [15].

| | Valor | |
|------|---|--|
| | 0 | 1 |
| CPOL | Linha inativa com tensão no valor baixo | Linha inativa com tensão no valor elevado |
| CPHA | Dados amostrados a partir da primeira transição do <i>clock</i> | Dados amostrados a partir da segunda transição do <i>clock</i> |

Deste modo é possível quatro diferentes modos de operação no SPI, que são explícitos na tabela 6.3.

Tabela 6.3: Diferentes modos de operação do SPI.

| Modo | CPOL | CPHA |
|------|------|------|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |

Um cartão SD requer o modo 0, ou seja, $CPOL = 0$ e $CPHA = 0$ e então a comunicação com estes componentes decorre como demonstrado na figura 6.4.

A *interface* com um cartão SD via protocolo SPI trata-se de uma comunicação comando-resposta em que todos os comandos são dados pelo *Master*. Os comandos SD são designados por $CMDXX$ e $ACMDXX$ em que CMD refere-se a comando e $ACMD$ a comando específico de aplicação. Estes são comandos são enviados em tramas de comandos cuja constituição é representada na tabela 6.4.

Tabela 6.4: Estrutura de tramas de comando[16].

| Byte 1 | | | Byte 2 – 5 | Byte 6 | |
|--------|---|---------------|------------|--------|---|
| 0 | 1 | Nº de Comando | Argumento | CRC | 1 |

Antes de se enviar o número do comando são enviados 2bits de início de trama. No argumento é enviado informação adicional sobre o comando. No final da trama é enviado um código CRC de 7bits seguido de um *stop bit*. Por predefinição o CRC é desprezado pelo cartão SD. Na tabela 6.5 registam-se alguns dos comandos mais recorrentes do modo SPI no protocolo SD.

Tabela 6.5: Alguns comandos SD [16].

| Comando | Argumento | Comprimento do argumento | Descrição |
|---------|-------------------------------|--------------------------|---|
| CMD0 | Nenhum | Nulo | Indicar o cartão a reiniciar e entrar no estado inativo |
| CMD16 | Comprimento do bloco de dados | 32 <i>bits</i> | Definir comprimento do bloco de dados |
| CMD17 | Endereço do bloco de dados | 32 <i>bits</i> | Ler um único bloco de dados |
| CMD24 | Nenhum | 32 <i>bits</i> | Escrever um único bloco de dados |
| CMD41 | Nenhum | Nulo | Inicializar cartão SD |
| CMD55 | Nenhum | Nulo | Próximo comando será específico de aplicação (ACMDXX) |

Após um comando, o *Slave* envia a trama de resposta contendo um *token* dependendo do comando notificando o início de uma transmissão de dados (caso tenham sido requeridos dados) ou uma condição de erro. Após este *token* do pacote de dados, são enviados os dados do bloco propriamente dito. A estrutura de um pacote de dados é demonstrada na tabela 6.6.

Tabela 6.6: Pacote de dados de uma comunicação com cartão SD no modo SPI [17].

| | <i>Token</i> de dados | Bloco de dados | CRC |
|--------------------------|-----------------------|----------------|-----|
| Tamanho [<i>bytes</i>] | 1 | de 1 a 2048 | 2 |

6.2.3 Protocolos SD

O modo SD incorpora dois diferentes protocolos o SD 1 – *bit* e o SD 4 – *bit*. A maior diferença entre estes dois protocolos reside no facto de o primeiro usar uma linha de dados enquanto que o segundo usa quatro conseguindo assim taxas de transmissão até quatro vezes superiores.

Tal como no SPI é usada a linha *clock* para sincronização. Este modo permite que diferentes cartões SD partilhem o mesmo barramento. A linha de comandos destina-se ao envio de tramas que iniciam a comunicação, estes comandos pode ser destinados a todos os *slaves*/cartões (*broadcast*) ou para um único cartão (*Unicast*). Os diferentes comandos enviados estão sujeitos ao CRC de 7 *bits*.

Os dados transmitidos são protegidos pelo CRC de 16 *bits*, que é calculado e enviado para cada linha em particular [16].

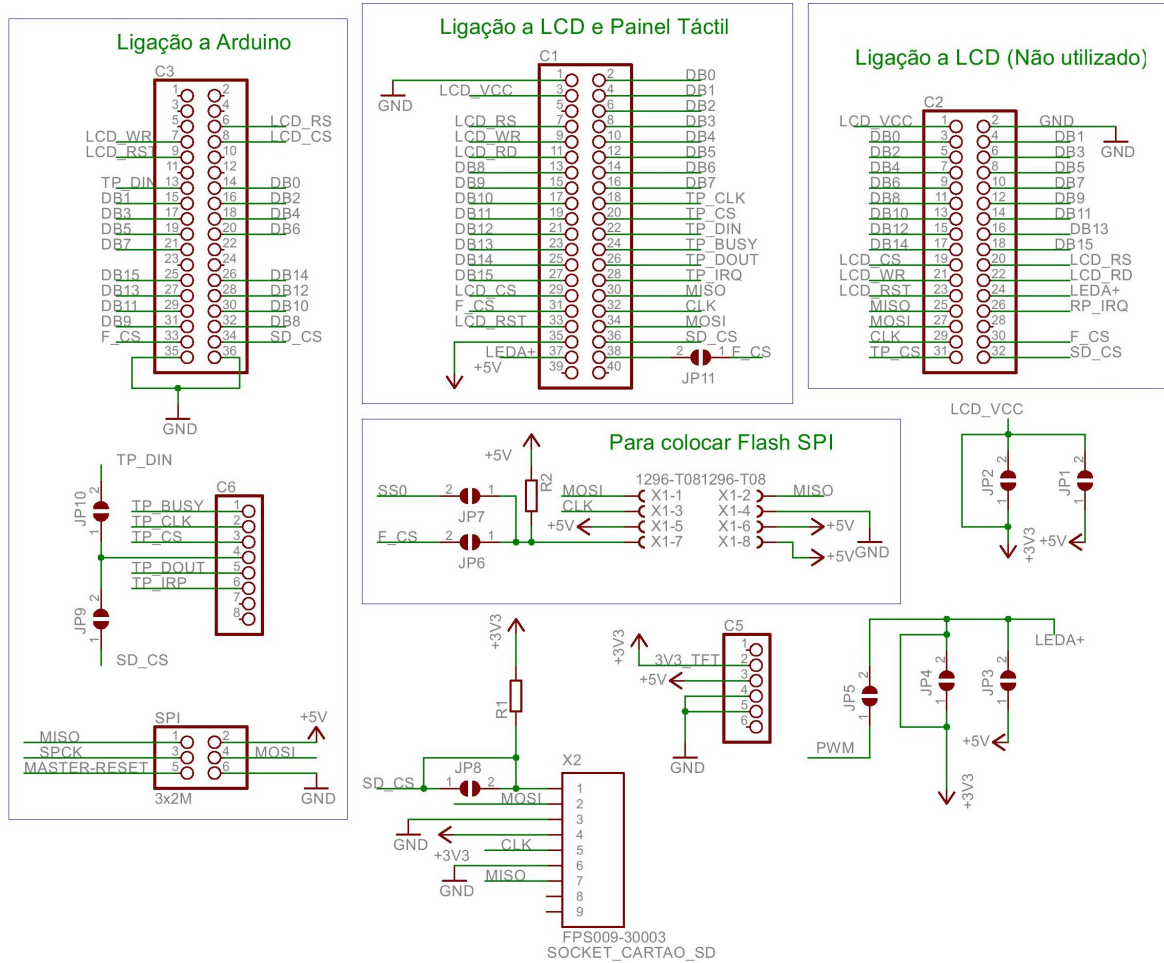
6.2.4 Protocolo utilizado

O protocolo de comunicação implementado foi o SPI visto ser o mais apropriado para MCUs e existir bibliotecas no IDE Arduino que permitem uma comunicação SPI assim como a manipulação de ficheiros em um cartão SD através de uma comunicação SPI. A frequência da linha de *clock* é de 4 *MHz*.

6.2.5 Ligações para cartão SD, LCD e painel tátil

O cartão SD é colocado numa ranhura existente no *shield* de ligação entre Arduino e módulo LCD, que se pode observar na figura 3.4 da subsecção 3.2.4. Como referido na mesma subsecção, o *shield* para além de se destinar a efetuar as ligações entre módulo LCD possibilita uma comunicação SPI entre cartão SD e Arduino. Na figura 6.5 demonstra-se o esquema elétrico do *shield*.

No mesmo *shield* realizaram-se certas ligações por intermédio de *jumpers* existentes nesta placa de forma a configurá-la para as funções pretendidas. Neste sentido soldou-se o *jumper* 8 para habilitar o *Chip Select* do cartão SD. As ligações elétricas relativas ao LCD e respetivo painel tátil serão referidas novamente no capítulo 7.

Figura 6.5: Esquema elétrico do *shield* LCD.

6.3 Real Time Clock

O MCU SAM3X8E, incluído no Arduino Due, possui um periférico de RTC com suporte de calendário. Com este periférico consegue-se contar tempo com precisão em unidade reais (segundos, minutos, horas, dias, meses e aos) conseguindo um registo e controlo precisos sobre o tempo e data. O calendário e relógio (RTC) foram implementados paReRra assim os registos energéticos poderem ser localizados no tempo, para além disso é adicionada uma certa comodidade ao utilizador poder consultar a data e hora na *Interface* como exemplificado na figura 7.4.

Este periférico recorre a um oscilador externo de 32,768 kHz . Tal como o CAN, o periférico RTC não tem atribuída nenhuma biblioteca oficial o que motivou o uso de uma biblioteca desenvolvida por uma entidade individual e publicada *online* [48].

A data e hora atuais são inseridas pela *Interface*, e a seguir é ativado e configurado o RTC, inicializada a comunicação SPI e criado um novo ficheiro de texto para registo dos próximos balanços energéticos. Esta última questão será abordada mais detalhadamente na secção

6.5, enquanto que a componente da *Interface* dedicada à configuração do RTC é abordada posteriormente na subsecção 7.4.1.

Antes de configurado o RTC a data e hora são validados pela biblioteca RTC, por exemplo o mês inserido é limitado entre 1 e 12, caso ultrapasse estes valores é assumido um valor predefinido e o mesmo se aplica à hora e minuto. No entanto a validação do dia não tem em conta o número de dias do mês específico. Para melhorar este aspeto antes de se configurar o RTC, realiza-se a validação do número do dia tendo em conta o mês e ano associado o que implica determinar se o ano em questão é bissexto. Deste modo recorreu-se a um método disponível na *Application Programming Interface* (API) da biblioteca RTC que indica se um determinado ano é bissexto, e deste modo calcula-se o número de dias de cada mês e então valida-se o dia digitado no painel tátil.

6.4 Cálculo de consumos e regeneração energética

Sempre que recebida uma mensagem CAN contendo o valor da tensão ou da corrente no motor BLDC, é incrementada a energia consumida ou regenerada no último segundo.

Para calcular a energia consumida/regenerada é necessário conhecer o valor do intervalo de tempo decorrido. A energia é calculada em *Joules*, pela fórmula da expressão 6.1.

$$Energia [J] = Tensão [V] * Intensidade [A] * Tempo [s] \quad (6.1)$$

Para obter valores de intervalos de tempo recorreu-se ao periférico do SAM3X8E, *Timer Counter* 1 que foi configurado manualmente não recorrendo a bibliotecas já existentes. Este periférico permite o incremento de um registo interno a uma frequência conhecida, e assim permitindo adquirir o valor de um intervalo de tempo. Os periféricos *Timer Counter* permitem igualmente gerar interrupções no MCU, e o *Timer Counter* 2 foi configurado para causar interrupções de um em um segundo.

As circunstâncias em que se efetua o balanço energético e respetivos registo é esquematizado no fluxograma da figura 6.6. Para além das circunstâncias são demonstrados subprocessos como cálculos.

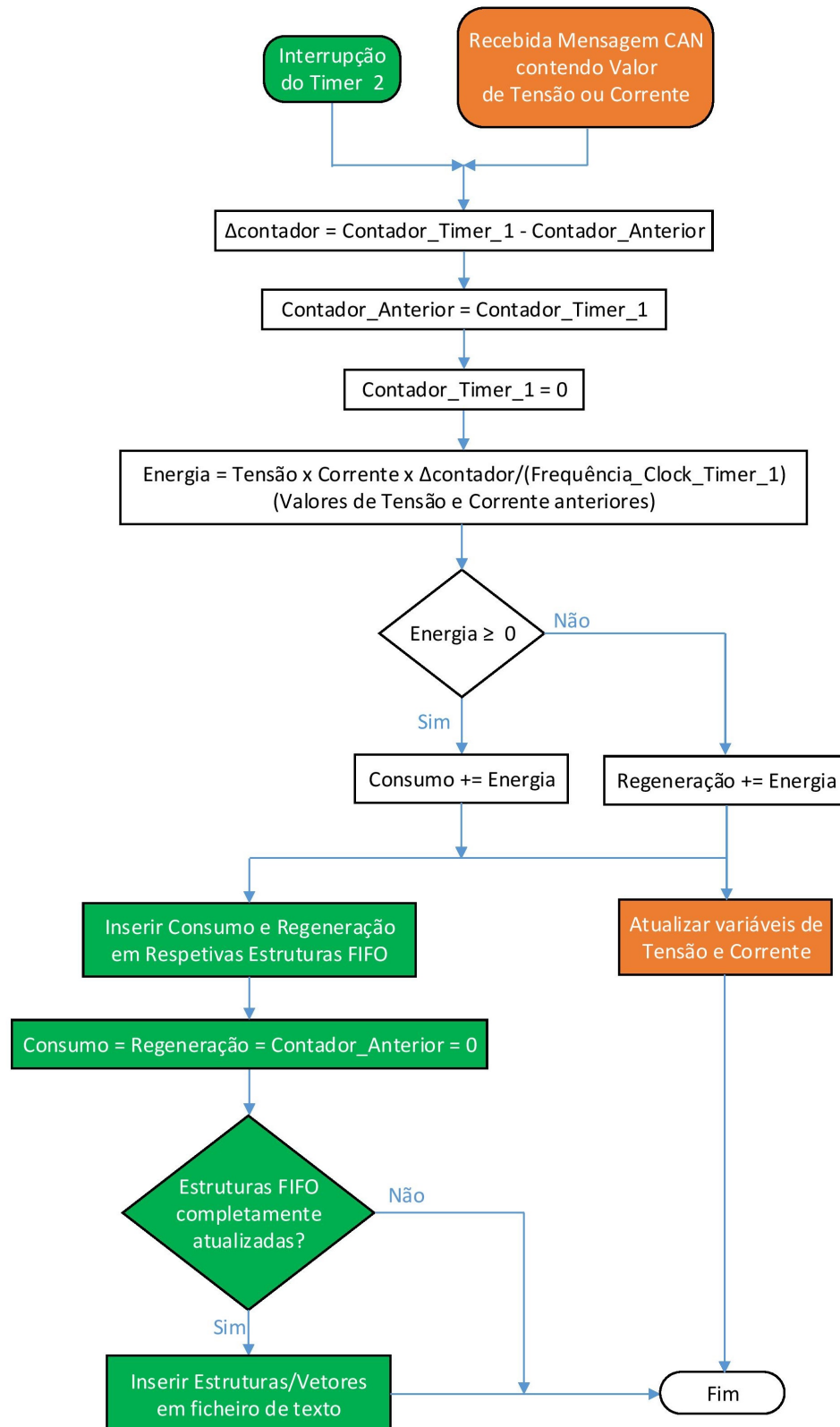


Figura 6.6: Fluxograma de cálculo de balanços energéticos e respetivo registo.

A função de cálculo de consumo ou regeneração é executada se:

- Recebida uma mensagem CAN contendo o valor da tensão ou corrente no motor BLDC;
- Ocorrer uma interrupção relativa ao *Timer Counter 2*.

A fórmula de cálculo de energia referente ao intervalo de tempo entre execuções da função de cálculo é demonstrada na figura anterior. O intervalo de tempo é a diferença entre valor do contador do *Timer* (“Contador_Timer_1” na figura 6.6) da execução atual com a anterior, dividida pela frequência com que este contador é incrementado. Para se saber a diferença entre contador do *Timer* execução atual com a anterior recorre-se à variável “Contador_Anterior”.

Se a função anterior ser corrida pelo primeiro motivo (mensagem CAN), os processos destacados a cor de laranja são executados, caso seja o segundo motivo (interrupção do *Timer*) os processos a verde são executados.

Se o motivo da execução da função anterior ser a primeira referida, após o cálculo energético as variáveis internas do SAM3X8E referentes à tensão e corrente são atualizadas.

Neste MCU são armazenados dois vetores, com as mesmas dimensões, contendo um os consumos energéticos e outro a regeneração energética dos últimos segundos. Estes vetores formam estruturas de dados *First In First Out* (FIFO) e são atualizados cada vez que é executada a função de interrupção, de modo a que cada posição dos vetores corresponde ao consumo/regeneração de um intervalo de um segundo.

Sempre que estas estruturas são totalmente preenchidas, os registos energéticos contidos nestes vetores são registados no cartão SD. Este processo é descrito mais detalhadamente na secção 6.5. Para além de servirem como *buffers* para registo de consumo/regeneração no cartão SD, estes vetores são utilizados ainda para se esboçar gráficos na *Interface* desenvolvida.

6.5 Registo nos ficheiros de texto

Os ficheiros são registados com o nome “CERP_#X.txt” em que “X” corresponde ao número do ficheiro. Toda a gestão de ficheiros e escrita de conteúdo é esquematizada na figura 6.7.

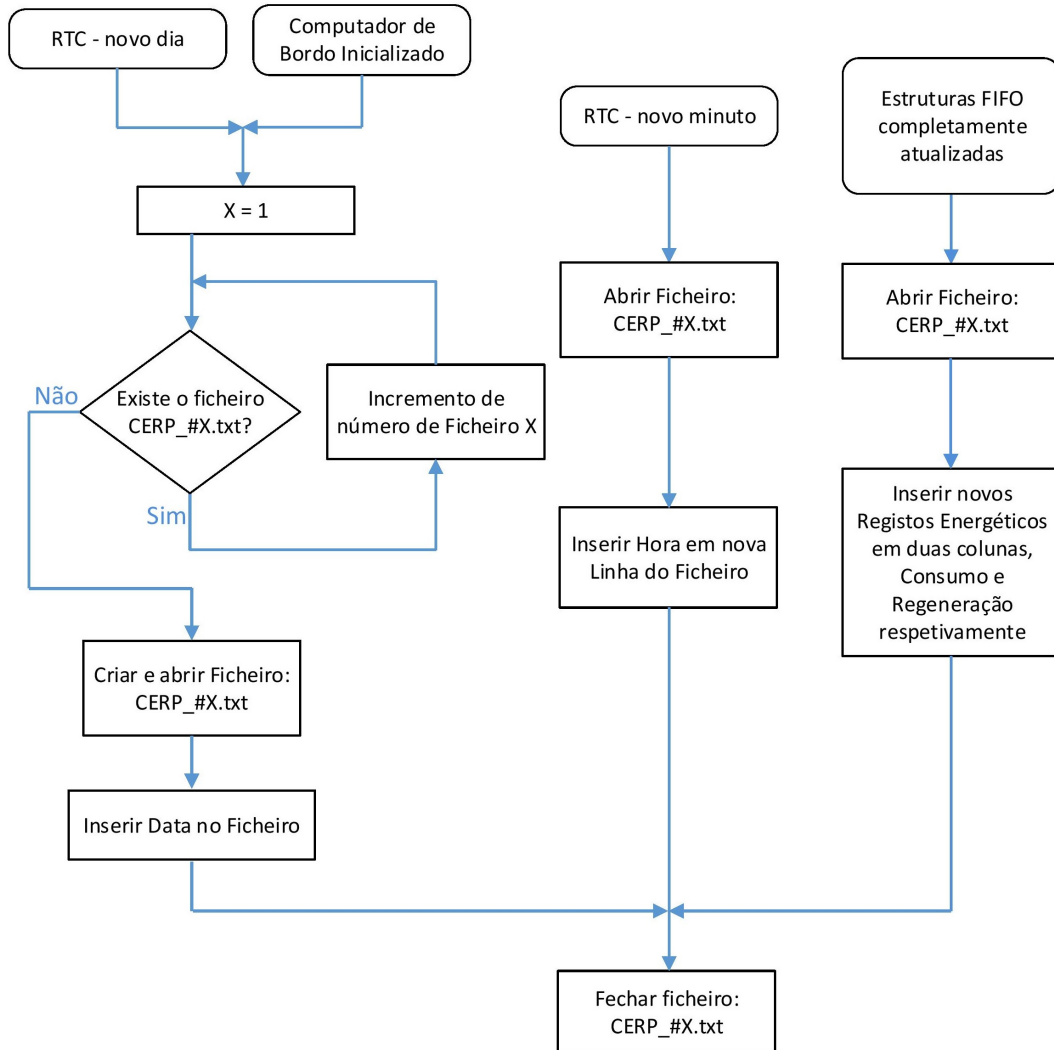


Figura 6.7: Fluxograma de gestão de ficheiros de texto e respetiva escrita.

Sempre que é recebido uma mensagem via CAN que indicando que a UC foi desligada, os registos energéticos são interrompidos. Caso o Computador de Bordo seja informado que a UC tenha sido ligada, é inserida no ficheiro uma nova linha indicar a hora em que os registos são retomados. Se a UC for ligada pela primeira vez num dia é criado um novo ficheiro. Caso o número de ficheiros atingir o limite de 99, todos os ficheiros de texto são eliminados e o contador “X” é reiniciado.

Na execução da rotina de interrupção acima referida, caso os vetores (estruturas FIFO) tenham sido totalmente atualizados, o ficheiro de texto mais recente é reaberto e nas seguintes linhas registam-se o consumo (esquerda) e regeneração (direita) em *Joules* dos últimos segundos. Deste modo no ficheiro são inseridas duas colunas, uma para consumo e outra para regeneração como se verifica na figura 6.8.

Na mesma rotina é igualmente atualizado o relógio no ecrã e se for caso, a data. Se houver uma transição de minuto é inserida no ficheiro uma linha com a hora e minuto atual, para mais facilmente se localizar no tempo os registos. Caso o haja uma transição de dia é criado

um novo ficheiro e a primeira linha contém o dia a que esse ficheiro se refere. A figura 6.8 exemplifica o conteúdo de um ficheiro de texto criado pelo Computador de Bordo.

| CERP #1.txt | |
|-------------|---|
| 22/05/2015 | |
| 15:58:01 | |
| 0 | 0 |
| 3 | 1 |
| 5 | 4 |
| 1 | 3 |
| 5 | 0 |
| 20 | 0 |
| 28 | 0 |
| 93 | 0 |
| 87 | 0 |
| 157 | 0 |
| 172 | 0 |
| 175 | 0 |
| ... | |
| 125 | 0 |
| 146 | 0 |
| 194 | 0 |
| 15:59:00 | |
| 184 | 0 |
| 223 | 0 |
| 228 | 0 |
| ... | |
| 297 | 0 |
| 141 | 0 |
| 16 | 0 |
| 0 | 3 |
| 6 | 0 |
| 2 | 2 |
| 6 | 3 |
| 16:00:00 | |

Dia a que se refere o ficheiro de texto
 Minuto a que se referem os próximos registos
 Registos de cada segundo
 Consumo —
 Regeneração —

Figura 6.8: Exemplo de registos energéticos em ficheiro.

O texto a escrever no ficheiro está contido em objetos da classe *String* disponível no IDE do Arduino. Apesar consumir mais memória que simples vetores de caracteres, esta classe é usada para conseguir simples conversões de inteiro para *String* e concatenações.

Capítulo 7

Computador de Bordo - *Interface*

Neste capítulo será mencionado o trabalho desenvolvido sobre o Computador de Bordo no que toca à sua *Interface* entre utilizador e computador, demonstrando como se utiliza. Isto implica demonstração das capacidades desta *Interface* e resultados obtidos com estas capacidades. Esta *Interface* trata-se de uma *Graphical User Interface* (GUI) e serão ainda descritos os recursos e métodos de desenvolvimento desta.

7.1 Técnicas de desenvolvimento da componente visual da *Interface*

A biblioteca utilizada para controlo do TFT designada é UTFT, já mencionada na subsecção 3.2.4. Esta foi inserida no IDE do Arduino.

Esta biblioteca orientada a objetos em que o LCD é o objeto em questão, permite o desenho de figuras geométricas elementares (círculos, retângulos e linhas), imprimir texto como *String*, número inteiro e número decimal. A API, incluída nesta biblioteca contém métodos/funções que cumprem as funções referidas anteriormente e exigem nos seus argumentos a localização (coordenadas) do texto ou forma geométrica a esboçar no ecrã em pixels. Antes de se desenhar uma forma ou imprimir texto é necessário definir a cor desejada e para tal recorre-se a uma função que define a cor atual no sistema *Red Green Blue* (RGB). Todo o texto, tabelas, gráficos e figuras demonstradas pela *Interface* foram concebidas a partir desta API. O texto pode ser ilustrado em diferentes tipos de letra. Na expressão 7.1 exemplifica-se um exemplo de função fornecida pela API em questão, que neste caso desenha um retângulo opaco.

$$UTFT :: fillRect(int X1, int X2, int Y1, int Y2) \quad (7.1)$$

As funções da biblioteca UTFT que imprimem texto e desenhavam figuras geométricas, implicam o envio sistemático (uma vez para cada pixel em questão) de comandos e dados para o controlador SSD1963, para que se defina qual as coordenadas do pixel a alterar assim como

o valor deste no sistema de cores RGB. Esta biblioteca internamente calcula qual a região de interesse, ou seja quais os pixels a ajustar conforme a operação de desenho pretendida.

Outra capacidade da mesma biblioteca utilizada é a demonstração de figuras, estando estas em *bitmaps* por si armazenados como vetores de variáveis no código do Arduino, ou seja na memória *flash*. Para isso o autor desta biblioteca fornece uma aplicação que converte imagens em vetores compatíveis [40]. O nome da aplicação é ImageConverter 565 e sua *interface* com o utilizador demonstra-se na figura 7.1.

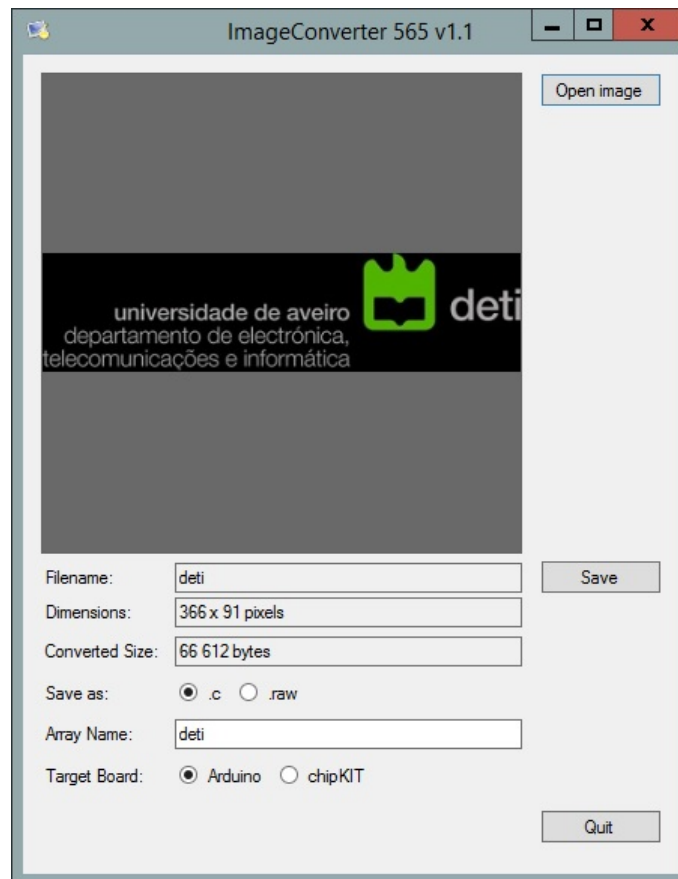


Figura 7.1: *Interface do software de conversão de imagens.*

7.2 Ligações elétricas do LCD e respetivas funções

Soldaram-se os *jumpers* 2 e 4 do *shield*, como se verificou nas figuras 5.4 e 6.5 para alimentar módulo TFT LCD a 3,3V, assim como alimentar o sistema de iluminação de fundo. Na tabela 7.1 alistam-se as linhas de alimentação e sinais da ligação entre Arduino e módulo TFT LCD.

Tabela 7.1: Sinais e alimentação LCD.

| Designação nos esquemáticos do <i>shield</i> e Arduino | Função | Designação do pino do controlador |
|--|------------------------------------|-----------------------------------|
| DBX : $X \geq 0 \wedge X \leq 15$ | Linhas de comandos e dados | DX |
| LCD_RST | Reiniciar LCD | RESET# |
| LCD_RS | Selecionar entre comandos e dados | D/C# |
| LCD_WR | Ciclos de escrita | WR# |
| LCD_RD | Ciclos de leitura | RD# |
| LEDA+ | Alimentação da luz de fundo do LCD | - |
| LCD_VCC | Alimentação do LCD | VDDIO e VDDLCD |
| LCD_CS | <i>Chip Select</i> | CS# |

A linha LCD_RS indica de transmissão de dados caso tenha o valor lógico 1, caso contrário são transmitidos comandos. Transições sinal LCD_WR indicam ciclos de escrita no LCD na mesma forma que os ciclos de leitura são definidos pelo sinal LCD_RD [38].

7.3 Desenvolvimento da componente tátil

De seguida é explicado o trabalho em torno do painel tátil, tanto a nível de *hardware* como *software*.

7.3.1 *Hardware*

O fabricante do LCD tátil não fornece os circuitos elétricos das implementações dos controladores do LCD e painel tátil. No entanto a ficha de dados do controlador do painel tátil XPT2046 exemplifica circuitos elétricos nos quais se utiliza este controlador. No anexo B encontra-se um extrato desta ficha de dados onde é explicado o funcionamento interno deste controlador ainda é demonstrado um circuito para utilização do XPT2046 [39].

7.3.2 *Software*

Para o utilizador explorar a maioria das diferentes funções da *Interface*, explicadas a seguir, existe no lado esquerdo do ecrã uma coluna de três botões tácteis, como se verifica nas figuras 7.4, 7.5, 7.7 e 7.9. Cada botão internamente (no código) corresponde uma estrutura de

dados desenvolvida propositadamente para a presente *Interface*. Os membros desta estrutura são:

- Texto;
- Cor do texto;
- Cor do fundo;
- Posição dos quatro extremos do botão - Coordenadas em X e Y.

Para além dos botões anteriores desenvolveu-se um teclado numérico tátil que é demonstrado na subsecção 7.4.1.

A API da biblioteca utilizada para o painel tátil e respetivo controlador, UTouch, permite que se detete facilmente se existem dados devido a contacto no painel e quais as coordenadas de contacto.

As ligações elétricas no Arduino Due referentes ao painel tátil podem ser constatadas nas figuras 5.4 e 6.5 e as linhas associadas a esta comunicação são as, TP_CLK, TP_CS, TP_DOUT, TP_DIN, TP_IRQ e TP_BUSY. TP significa *Touch Panel*¹. Verificam-se as habituais linhas de sinal do SPI TP_DIN e TP_DOUT que são as entradas e saídas SPI do controlador do painel tátil XPT2046, assim como o *Clock* e CS. Verifica-se a linha *Interrupt Request* (IRQ) para o SAM3X8E detetar se existem dados sobre contacto no painel, e ainda se verifica linha a *Busy* que está ativa enquanto qualquer linha transmissão (TP_DOUT ou TP_DIN) é usada.

Sempre que há registo de contacto no painel tátil, as coordenadas adquiridas são comparadas com todos os botões existentes. A leitura e processamento dos dados adquiridos pelo painel tátil é realizada no ciclo principal do Arduino Due, como demonstrado no respetivo fluxograma na figura 7.2.

¹ou Painel Tátil

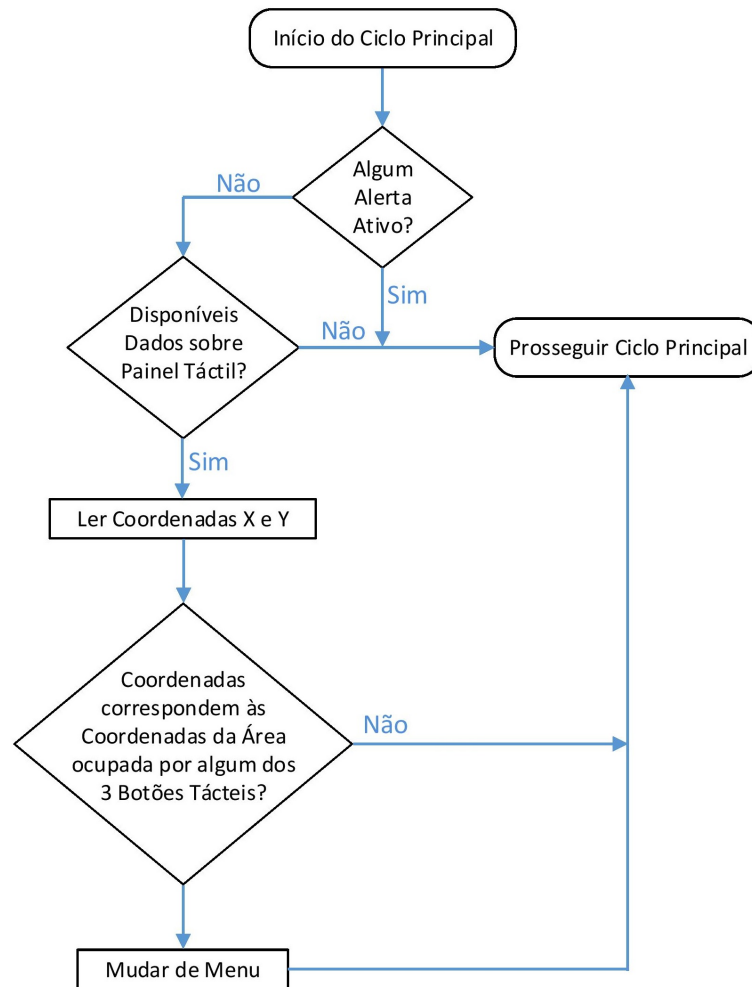


Figura 7.2: Fluxograma sobre leitura e processamento de dados adquiridos pelo painel táctil.

7.4 Funções da *Interface* desenvolvida e botões tácteis

As funções disponíveis são:

1. Configuração de data e hora, que ocorre na inicialização do Computador de Bordo;
2. Ver parâmetros em tempo real;
3. Ver registos recentes de consumo e regeneração;
4. Modo de baixo consumo do ecrã;
5. Informação acerca protótipo desenvolvido;
6. Alertas de problemas no veículo.

As funções da *Interface* número 2, 3, 4 e 6 são acedidas pelos botões demonstrados e numerados mais adiante na figura 7.4. A correspondência entre botões e funções da *Interface* é descrita na tabela 7.2.

Tabela 7.2: Correspondência entre botões e funções da *Interface*.

| Nº de Botão | Texto no Botão | Funções da <i>Interface</i> acedidas |
|-------------|---|--------------------------------------|
| 1 | “Ver variaveis em tempo real” | 2 |
| 1 | “Ver consumos no automovel” | 3 |
| 2 | “Limpar ecra” | 4 |
| 3 | “Informacao sobre prototipo desenvolvido” | 5 |

Por questões de espaço, o botão 1 serve para navegar entre duas diferentes funções.

7.4.1 Teclado numérico para configuração de data e hora

Quando o Computador de Bordo é ligado, é mostrado no ecrã um teclado numérico e é impresso texto solicitar que se digite a data e hora atual tal como demonstrado na figura 7.3.



Figura 7.3: Teclado matricial.

Os algarismos do teclado têm o mesmo formato que os algarismos dos *displays* de 7 segmentos mostrados na figura anterior, pertencem a um tipo de letra incluído na biblioteca UTFT. Esta função, que consiste em digitar no ecrã a data e hora atuais. Estes botões tratam-se de uma diferente de estrutura de dados cujos membros são as coordenadas dos extremos do botão. O teclado corresponde a um vetor de estruturas com dez elementos, e o índice corresponde ao valor numérico do botão respetivo. Durante a definição de data e hora, se houver contacto no painel táctil as coordenadas adquiridas são comparadas com as dos 10 botões tácteis, constituindo assim um processo semelhante ao descrito na subsecção 7.3.

Como já referido na secção 6.3 após digitada a data, é inicializado e configurado o RTC. Após digitadas a data e hora, todas as restantes funções da *Interface* são disponibilizadas.

7.4.2 Ver variáveis em tempo real

A componente *On-board diagnostics* da dissertação estabelece-se nesta função. Na figura 7.4 ilustra-se um exemplo prático da utilização desta função do Computador de Bordo. Pode-se igualmente verificar a coluna de botões anteriormente referida na parte esquerda do ecrã. Neste mesma figura foram destacados os três botões tácteis e o relógio explicado anteriormente na secção 6.3.

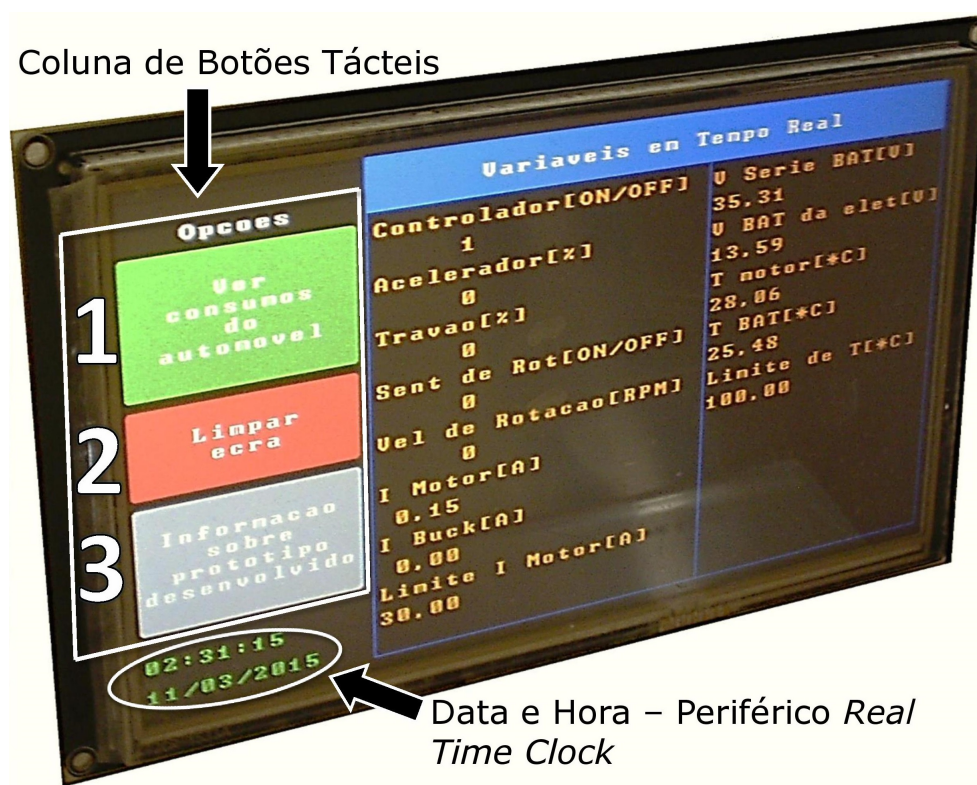


Figura 7.4: Menu de visualização de parâmetros em tempo real.

Na tabela do LCD é impresso o nome de cada variável/parâmetro CAN e logo abaixo o respetivo valor. Estes nomes estão armazenados em vetores de caracteres. As coordenadas da posição de cada nome de parâmetro e valor numérico associado, são calculados a partir do ID CAN da variável/parâmetro em questão.

Estes valores adquiridos pelos diferentes sensores são ilustrados em tempo real (sempre em atualização), o que permite detetar certos problemas como má calibração destes sensores. Permite ainda a deteção de mau processamento, por parte do Controlador do motor BLDC, dos valores adquiridos por estes mesmos sensores. Uma análise crítica destes parâmetros permite a deteção de vários exemplos de falhas.

Por exemplo: valores de corrente elevados estando os potenciômetros de acelerador e travão a zero, indicam falhas como curto-circuitos no *chopper*. Um exemplo de mau processamento dos sinais de saída de sensores, é o mau cálculo da velocidade de rotação resultando em velocidades erradas mesmo não havendo erro na leitura e ligação elétrica destes sensores. As temperaturas podem indicar problemas de sobreaquecimento. Caso este Computador de Bordo se aplicasse a um sistema de controlo de carro elétrico mais complexo, as temperaturas medidas e ilustradas na *Interface* poderiam indicar problemas na refrigeração.

7.4.3 Ver registos recentes de consumo e regeneração de experiências associadas

Se o botão 1 é premido enquanto a *Interface* demonstra os parâmetros do Controlador, ou seja quando este botão contem o texto “Ver consumos do automovel”, é executada a funcionalidade que consiste na visualização do balanço energético dos últimos segundos por meios gráficos. O balanço é calculado e demonstrado em *Joules*. Um exemplo de gráfico ilustrado pela *Interface* verifica-se na figura 7.5.

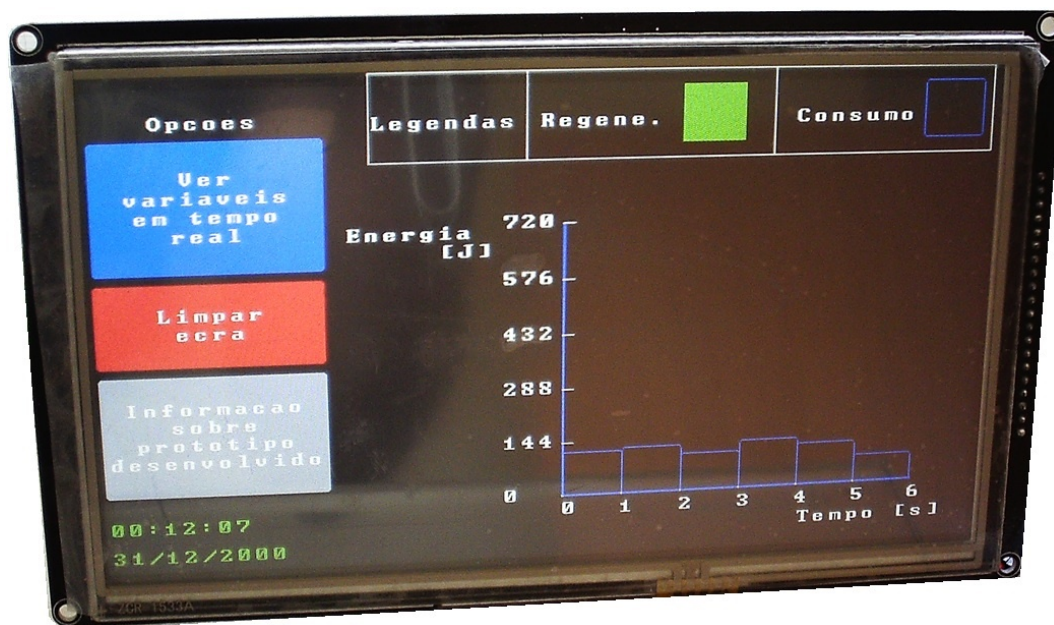


Figura 7.5: Menu de balanços energéticos, realizados durante uma fase de consumo.

O intervalo de tempo mais à esquerda é o mais recente. Nestes gráficos são representados todos o valores registados nos vetores de consumo e regeneração. Sempre que estas estruturas/vetores são completamente atualizadas o gráfico de balanços energéticos é atualizado.

O gráfico acima demonstrado foi construído recorrendo às funções da API destinadas ao desenho de retângulos e linhas. A posição horizontal de cada barra/retângulo é calculada conforme o índice/intervalo de tempo respetivo do vetor dos consumos ou regeneração, en-

quanto que a altura da cada barra é calculada conforme o valor do consumo/regeneração do respetivo intervalo de tempo/índice.

O mesmo gráfico foi realizado durante uma fase de consumo, em que motor BLDC tinha uma velocidade rotação de 1400 *RPM* e estava acoplado a um motor de indução trifásico que oferecia a este resistência mecânica. O motor trifásico neste ensaio não se encontrava eletricamente acionado, servindo apenas para oferecer resistência pela ligação mecânica entre motores.

Este acoplamento e o banco de baterias utilizado verifica-se na fotografia da figura 7.6, verifica-se que 2 baterias do banco não são utilizadas.

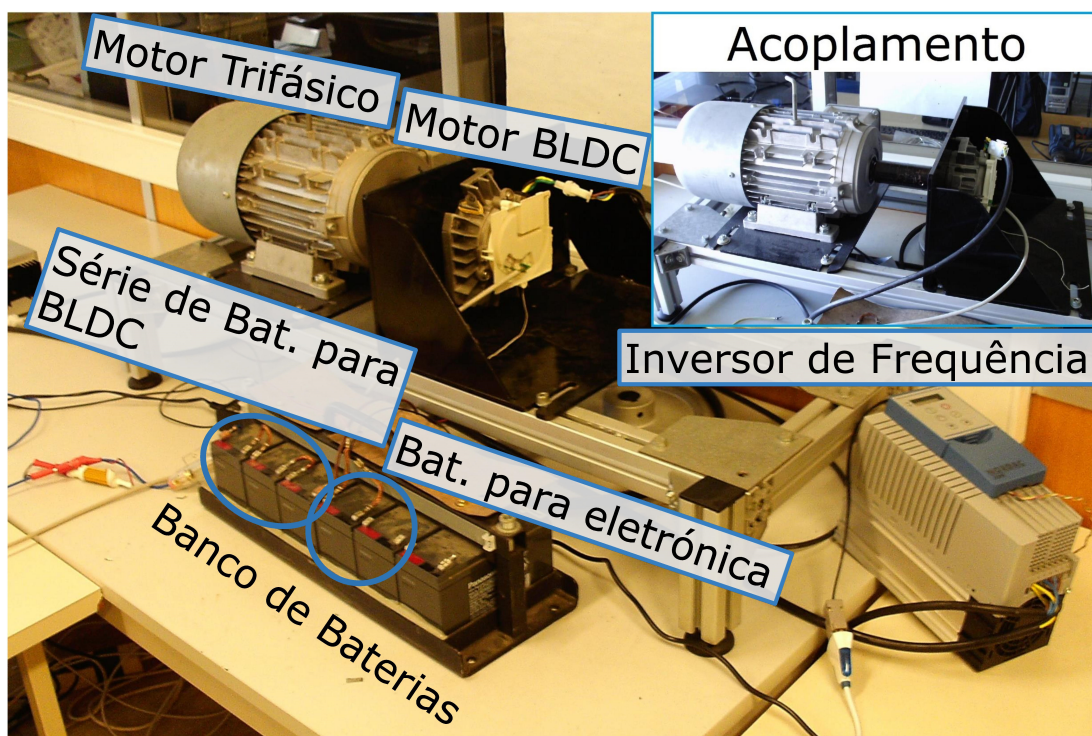


Figura 7.6: Acoplamentos entre motores para ensaios de consumo e regeneração.

Realizaram-se igualmente ensaios de regeneração de forma a adquirir gráficos de regeneração com o motor BLDC igualmente acoplado ao motor de indução trifásico, embora que nesta situação o motor trifásico entrega energia mecânica ao motor BLDC. Para controlar o motor trifásico com 3 *kW* de potência nominal, recorreu-se ao inversor de frequência NORDAC SK 520 E, que foi configurado para conseguir uma velocidade de rotação de 1200 *RPM* e a presença deste inversor pode-se verificar na figura 7.6. Para controlar o inversor de frequência utilizou-se o *software* NORD CON e a comunicação entre PC e inversor funcionava pela *interface* RS485.

Na UC colocou-se o travão a 90 %, visto após várias experiências ter-se concluído que o motor absorve maiores potências quando o *duty cycle* na ponte H é de 90 %. O resultado representa-se na figura 7.7.



Figura 7.7: Gráfico adquirido durante travagem regenerativa.

Nos exemplos de gráficos adquiridos apenas se representam intervalos de tempo que correspondem apenas a consumo ou regeneração. É impraticável conseguir consumo e regeneração em um intervalo de um segundo, no entanto se o código do Computador de Bordo fosse ajustado de forma a obter tempos de integração superiores haveriam intervalos em que ocorria regeneração e consumo. e nesse caso seriam ilustradas simultaneamente no LCD barras de regeneração e consumo. A barras de consumo são transparentes para não se sobreponem às barras de regeneração opacas que ficariam no interior das de consumo.

7.4.4 Alertas

Alguns parâmetros/variáveis que circulam na rede CAN podem ser analisados pelo próprio Computador de Bordo de forma a detetar anomalias. Certos parâmetros podem e devem ter valores limitados e a consequente excedência destes limites causa o acionamento de modos de alerta. Este limites podem ser verificados na tabela do menu “Ver variáveis em tempo real”, como se verifica na figura 7.4. De seguida abordam-se os diferentes alertas implementados pelo Computador de Bordo, que podem ocorrer em simultâneo. A interação com o utilizador é bloqueada de forma a que este não explore as outras funções do Computador de Bordo, até que todos parâmetros limitados assumam valores admissíveis. A figura 7.8 demonstra o aspeto da *Interface* caso ambos alertas sejam acionados. Estes alertas são descritos nas próximas subsecções.

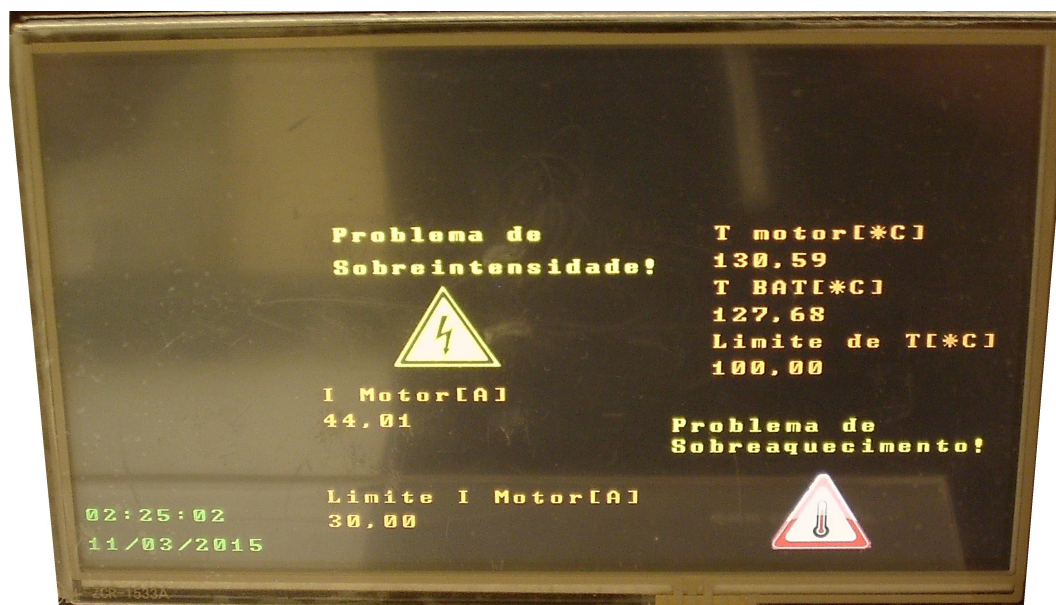


Figura 7.8: Exemplo de alerta de temperatura.

Sobreaquecimento

Como já referido no capítulo 5, a UC define um limite de temperatura para o motor e banco de baterias. Caso o Computador de Bordo detete que uma das temperaturas ultrapasse o limite, este mostra um alerta de sobreaquecimento, como se demonstra na figura 7.8. Os parâmetros do Controlador do motor demonstrados são o limite de temperatura

e quais temperaturas irregulares. Para além destes valores é ilustrado um ícone elucidativo a temperaturas elevadas.

Sobreintensidade

O controlador de motor internamente possui um limite de corrente elétrica, e caso este seja ultrapassado o sinal PWM enviado para a ponte H é desativado. Na *Interface* é gerado um alerta em que se mostra o atual valor da corrente elétrica atual e o respetivo limite. Analogamente ao alerta anterior é mostrada uma imagem que simboliza problemas elétricos, como se verifica na figura anterior.

7.4.5 Informação acerca todo o protótipo desenvolvido

Este menu contém texto que informa o utilizador acerca do protótipo desenvolvido como detalhes como os seus autores e em que âmbito foram desenvolvidos. Ainda mostra uma figura com o logótipo da Universidade e o nome do Departamento em que se desenvolveu a presente dissertação. A fotografia da figura 7.9 mostra o conteúdo deste menu.



Figura 7.9: Menu de informação sobre protótipo desenvolvido.

7.4.6 Limpar ecrã

Ao premir o botão com o texto “Limpar ecrã”, o ecrã entra no modo *sleep* conseguindo assim um menor consumo energético no ecrã. Para conseguir este modo a biblioteca UTFT foi ajustada visto esta não possuir um método público que permita acesso a este modo. Para tal foi necessária a consulta da ficha de dados do controlador de TFT SSD1963, forma a conhecer quais os comandos a enviar para ativar e desativar o modo *sleep*, que são 0x10 e 0x11 respetivamente [38].

Com este modo ativo todo o conteúdo do ecrã é limpo, e este mesmo modo é desativado quando se voltar a tocar no ecrã.

7.5 Consumos energéticos de todo o sistema eletrónico

Como já referido na secção 4.1, toda a eletrónica, ou seja a UC e Computador de Bordo são alimentados a partir de uma bateria de 12 V. Após experiências laboratoriais em que se alimentou ambas unidades a partir de uma fonte configurada a 12 V, registou-se na tabela 7.3 as potências consumidas em duas diferentes circunstâncias.

Tabela 7.3: Consumos energéticos da UC e Computador de Bordo.

| Modo de funcionamento | Tensão [V] | Corrente [mA] | Potência [W] |
|--------------------------|------------|---------------|--------------|
| Normal | 12 | 790 | 9,48 |
| LCD em modo <i>Sleep</i> | | 716 | 8,59 |

As potências elétricas referidas anteriormente são fornecidas pela bateria de 12 V que alimenta a eletrônica.

Capítulo 8

Conclusões e trabalho futuro

Neste capítulo é feita uma apreciação global do trabalho realizado, tendo em conta os resultados obtidos. No final deste capítulo são apresentadas propostas de trabalho futuro.

8.1 Conclusões

Com o estudo e reconstrução do Controlador do motor BLDC notou-se a necessidade de acrescentar componentes, como sensores de temperatura, interruptores de modo a obter mais parâmetros para registar e demonstrar o Computador de Bordo, para além tornar o modelo de UC existente mais próximo de uma ECU de um carro elétrico real.

A implementação dos sensores de temperatura teve um propósito experimental não conseguindo medir de forma adequada as temperaturas o motor e banco de baterias. Na realidade a montagem de sensores de temperatura teria de ser complexa em que o sensor de temperatura teria de ser colocado no interior do motor. Seria necessário realizar estudo sobre todo sistemas térmico envolvido para saber se estes sensores estariam sensíveis às temperaturas dos pontos críticos da estrutura.

O protocolo CAN permitiu uma simples e intuitiva troca de informação entre unidade de controlo e Computador de Bordo.

Os objetivos pretendidos com o Computador de Bordo, foram atingidos, visto este ilustrar em tempo real os principais parâmetros da UC, e ter a capacidade de calcular, registar consumos energéticos e ainda demonstrá-los de forma gráfica.

Os cálculos energéticos foram realizados de um modo destinado a experiências laboratoriais, em que os consumos energéticos são referentes a intervalos tempo na ordem dos segundos. Isto permitiu ensaios laboratoriais mais rápidos e práticos. Na realidade estes registos seriam mais lógicos se estes fossem referentes a intervalos de tempo mais longos, por exemplo 5 minutos e neste caso com uma unidade de energia diferente como o *kWh*. Para se concretizarem os ajustes anteriores é necessário ajustar o código do Computador de Bordo, obtendo intervalos de integração mais longos.

8.2 Trabalho futuro

Para um protótipo mais elaborado propõem-se as seguintes tarefas:

- Para um melhor rendimento energético, devem ser explorados os modos *Sleep* tanto na UC como no Computador de Bordo;
- Construir um conversor *Buck* de forma a garantir a carga da bateria de 12 V a partir da série de baterias;
- Realizar placas de circuito impresso¹ para a nova UC e placa de medidas;
- Desenvolver um método de estimativa da carga das baterias, e mostrar o estado destas na *Interface*;
- No Computador de Bordo incorporar as funções de *Cruise Control* disponíveis na UC definindo uma velocidade máxima no motor, com o sentido de acrescentar segurança a todo o sistema;
- Na comunicação CAN inserir gestão e manuseamento de erros em ambas as estações;
- No sentido de adquirir um protótipo de carro elétrico mais completo, desenvolver uma unidade de controlo de um sistema de transmissão que seja inserida na mesma rede CAN, esta unidade controlo poderia simplesmente simular o controlo de uma caixa de velocidades possibilitando o ajuste da razão de transmissão entre rotor do motor e rodas. Deste modo seria possível construir uma nova *interface* que corresponderia a um painel de instrumentos que indicasse por exemplo, a velocidade de rotação do motor, a velocidade do veículo e o estado de carga da bateria. Esta nova *interface* consistiria igualmente num LCD que representaria por exemplo velocímetros analógicos.

¹Ou *Printed Circuit Board* (PCB).

Bibliografia

- [1] R. E. Martins, *Construção De Um Carro Elétrico Puro: Instrumentação*, Departamento de Eletrónica, Telecomunicações e Informática, 2014.
- [2] A. Silva, “Redes de comunicações no automóvel,” *Sistemas Automóveis*, 2009. [Online]. Available: http://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos%202008-09/SIAUT2009_ComunicacoesAutomovel.pdf
- [3] J. C. Metrôlho, “Rede can para comando de actuadores em estufas agrícolas,” Master’s thesis, Universidade do Minho, 1999. [Online]. Available: <http://repositorio.ipcb.pt/bitstream/10400.11/497/1/MetrolhoMsc%20dissertacao%2099%5b1%5d.pdf>
- [4] I. I. o. T. Mr. Steve Talbot, Dr. Shangping Ren. Deeply embedded control systems, cyber physical systems (cps) in passenger vehicles: Survey of can, ttcan, flexray, lin. Website. Illinois Institute of Technology. Chicago.
- [5] P. D. Inc. (2014) Vehicle obd ii compatibility. Website. PLX Devices Inc. [Online]. Available: <https://plxdevices.com/obd/>
- [6] mytoolsforyou.com. (2014) Kts 340 base kit. Website. mytoolsforyou.com. [Online]. Available: <http://www.mytoolsforyou.com/automotive-tools/BSH-F00E900400.html>
- [7] B. Gunn, “Comprehensive experimental analyses of automotive attack surfaces,” in *USE-NIX Security*, 2011.
- [8] G. Hymes. (2008) Prius consumption monitor - 2008 toyota. Website. [Online]. Available: <https://www.flickr.com/photos/garyhymes/2424649836/>
- [9] D. Systems, “Picadillo-35t,” 4D Systems, Exposição, 2014. [Online]. Available: http://www.4dsystems.com.au/product/Picadillo_35T/
- [10] ——. (2014) 4d systems | ulcd-70dt. Website. 4D Systems. [Online]. Available: <http://www.4dsystems.com.au/product/uLCD-70DT/>
- [11] SainSmart. (2014) Sainsmart mega2560 + 7"7 inch tft lcd screen sd card slot + tft shield for arduino. Website. SainSmart. [Online]. Available: <http://www.sainsmart.com/sainsmart-due-7-7-inch-tft-lcd-screen-sd-card-slot-tft-shield-for-arduino-1.html>

- [12] ——. (2015) Sainsmart due + 7"7 inch tft lcd screen sd card slot + tft shield for arduino. Website. SainSmart. [Online]. Available: <http://www.sainsmart.com/sainsmart-due-7-7-inch-tft-lcd-screen-sd-card-slot-tft-shield-for-arduino.html>
- [13] M. Ltd. (2015, Março) Component: Fat (sd, sdhc) (storage). Website. Matrix Ltd. [Online]. Available: [http://www.matrixsl.com/wiki/index.php?title=Component:_FAT_\(SD,_SDHC\)_\(Storage\)](http://www.matrixsl.com/wiki/index.php?title=Component:_FAT_(SD,_SDHC)_(Storage))
- [14] byteparadigm.com. (2015) Introduction to i2c and spi protocols. byteparadigm.com. [Online]. Available: <http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/>
- [15] DLNWARE. (2015) Spi transfer modes. DLNWARE. [Online]. Available: <http://dlnware.com/theory/SPI-Transfer-Modes>
- [16] F. Foust, *Secure Digital Card Interface for the MSP430*, Michigan State University, 2004.
- [17] Chan. (2013, Fevereiro) How to use mmc/sdc. The Electronic Lives Manufacturing. [Online]. Available: http://elm-chan.org/docs/mmc/mmc_e.html
- [18] E. Schaal. (2014, Novembro) The 10 electric vehicles with the longest driving range. Website. Wall St. Cheat Sheet. [Online]. Available: <http://wallstcheatsheet.com/automobiles/top-10-electric-vehicles-with-the-longest-driving-range.html/?a=viewall>
- [19] T. Swan. (2014, Agosto) 2015 volkswagen e-golf. Website. Car and Driver. [Online]. Available: <http://www.caranddriver.com/volkswagen/e-golf>
- [20] MOBI.E. (2015) Pesquisa de postos. Website. MOBI.E. [Online]. Available: <http://www.mobie.pt/pt/postos-de-carregamento>
- [21] F. Sunderland. (2014) European electric car sales could reach one million by 2025. Website. Contract Hire and Leasing Deals. [Online]. Available: <http://www.contracthireandleasing.com/car-leasing-news/european-electric-car-sales-could-reach-one-million-by-2025/>
- [22] A. iQ. (2013, Junho) Automotive diagnostic systems: History of obd. Automotive iQ. [Online]. Available: <https://automotiveiq.wordpress.com/2011/06/08/automotive-diagnostic-systems-history-of-obd/>
- [23] A. Chong, "The growth of automotive electronics in apac, the next frontier," in *Driving Asia - As Automotive Electronics Transforms a Region*. Technologies Asia Pacific Pte Ltd, 2010, ch. 1.
- [24] G. Davies. (2003, Agosto) Making sense of the displays. [Online]. Available: <http://prius.ecrostech.com/original/Understanding/MakingSenseOfDisplays.htm>

- [25] M. M. das Neves Bento, “Projecto e construção de um controlador para motor de sem escovas,” Master’s thesis, Universidade de Aveiro - Departamento de Electrónica, Telecomunicações e Informática, 2013.
- [26] G. Miller. (2013, Março) Automotive communication protocols: Preparing for the future. Website. EECatalog. [Online]. Available: <http://eecatalog.com/automotive/2013/03/13/automotive-communication-protocols-preparing-for-the-future/>
- [27] S. of Automotive Engineers, *J2411 - Single Wire Can Network for Vehicle Applications*, Society of Automotive Engineers Std., 2 2002. [Online]. Available: http://standards.sae.org/j2411_200002/
- [28] C. in Automation (CiA). (2015) Can in automation (cia): Time-triggered can. CAN in Automation (CiA). [Online]. Available: <http://www.can-cia.org/index.php?id=166>
- [29] J. S. F. J. A. Cook, “Controller area network (can),” *EECS 461*, 2008.
- [30] R. B. GmbH, “Can specification version 2.0,” Tech. Rep., Setembro 1991.
- [31] ST, “Lin (local interconnect network) solutions,” STMicroelectronics, Tech. Rep., 2002. [Online]. Available: http://www.st.com/web/en/resource/technical/document/application_note/CD00004273.pdf
- [32] H. M. P. Santos, “Redes de comunicação automóveis,” *Sistemas Automóveis*, 2007.
- [33] I. S. Organization, *ISO 17458-1:2013*, International System Organization Std., Janeiro 2013. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=59804
- [34] W. Lawrenz, *CAN System Engineering: From Theory to Practical Applications*, 2nd ed., W. Lawrenz, Ed. Springer, Dezembro 2013.
- [35] S. of Automotive Engineers, *SAE J1979 2006 edition - Ballot*, Society of Automotive Engineers Std., 2006. [Online]. Available: http://suzuki-club.ru/attachments/75537/?temp_hash=3cc10ac7ead6c32b55fe1c1bd0e462bc
- [36] O. O. Facile. (2014) Obd modes and configuration (pid). Outils OBD Facile. [Online]. Available: <http://www.outilsobdfacile.com/obd-mode-pid.php>
- [37] Bosch, “Esi[tronic]2.0 - the new diagnostics software from bosch,” 2015.
- [38] S. SYSTECH, “Ssd1963 advance information,” Tech. Rep., Janeiro 2010.
- [39] S. X. TECHNOLOGY, “Xpt2046 data sheet,” Tech. Rep., 2007.
- [40] H. Karlsen. (2014) Rinky-dink eletrronics. Website. Rinky-Dink Eletrronics. [Online]. Available: <http://www.rinkydinkelectronics.com>

- [41] LEM, “Current transducer casr series,” Tech. Rep., Março 2012.
- [42] T. Instruments, “Lm35 precision centigrade temperature sensors,” Texas Instruments, Tech. Rep., Janeiro 2015.
- [43] —, “Fully-differential isolation amplifier (rev. c),” Texas Instruments, Tech. Rep., Setembro 2013.
- [44] Panasonic, “Valve-regulated lead acid batteries: Individual data sheet,” Tech. Rep., 2005. [Online]. Available: <http://datasheetz.com/data/Batteries/Lead%20Acid/P218-datasheetz.html>
- [45] S. Corrigan, “Introduction to the controller area network (can),” Texas Instruments, Tech. Rep., Julho 2008.
- [46] T. A. da Costa Gonçalves, “Implementação e testes de robustez do protocolo tempo-real ftt-can,” Master’s thesis, Universidade de Aveiro - Departamento de Electrónica, Telecomunicações e Informática, 2010.
- [47] C. Kidder. (2015, Janeiro) due can. GitHub. [Online]. Available: https://github.com/collin80/due_can
- [48] L. Miliani. (2014) advancedfunctions. GitHub. [Online]. Available: <https://github.com/leomil72/advancedFunctions>

Anexos

A Excerto da ficha de dados do AMC1200

AMC1200 AMC1200B



SBAS542C – APRIL 2011 – REVISED SEPTEMBER 2013

www.ti.com

PACKAGE CHARACTERISTICS⁽¹⁾

| PARAMETER | TEST CONDITIONS | MIN | TYP | MAX | UNIT |
|--|---|------------------|--------------------|-----|------|
| L(I01) Minimum air gap (clearance) | Shortest terminal to terminal distance through air | DWV package 7 | 8 | | mm |
| L(I02) Minimum external tracking (creepage) | Shortest terminal to terminal distance across the package surface | DWV package 7 | 8 | | mm |
| CTI Tracking resistance (comparative tracking index) | DIN IEC 60112/VDE 0303 part 1 | ≥ 400 | | | V |
| Minimum internal gap (internal clearance) | Distance through the insulation | 0.014 | | | mm |
| R _{IO} Isolation resistance | Input to output, V _{IO} = 500 V, all pins on each side of the barrier tied together to create a two-terminal device, T _A < +85 °C | | > 10 ¹² | | Ω |
| | Input to output, V _{IO} = 500 V, +85 °C ≤ T _A < T _A max | | > 10 ¹¹ | | Ω |
| C _{IO} Barrier capacitance input to output | V _I = 0.5 V _{PP} at 1 MHz | | 1.2 | | pF |
| C _I Input capacitance to ground | V _I = 0.5 V _{PP} at 1 MHz | | 3 | | pF |

(1) Creepage and clearance requirements should be applied according to the specific equipment isolation standards of a specific application. Care should be taken to maintain the creepage and clearance distance of the board design to ensure that the mounting pads of the isolator on the printed circuit board (PCB) do not reduce this distance. Creepage and clearance on a PCB become equal according to the measurement techniques shown in the [Isolation Glossary](#) section. Techniques such as inserting grooves and/or ribs on the PCB are used to help increase these specifications.

ELECTRICAL CHARACTERISTICS

All minimum/maximum specifications at T_A = –40 °C to +105 °C and within the specified voltage range, unless otherwise noted. Typical values are at T_A = +25 °C, VDD1 = 5 V, and VDD2 = 3.3 V.

| PARAMETER | | TEST CONDITIONS | AMC1200, AMC1200B | | | UNIT |
|--------------------|---------------------------------------|---|-------------------|--------|--------|-------------------|
| | | | MIN | TYP | MAX | |
| INPUT | | | | | | |
| | Maximum input voltage before clipping | VINP – VINN | ±320 | | | mV |
| | Differential input voltage | VINP – VINN | –250 | | +250 | mV |
| V _{CM} | Common-mode operating range | | –0.16 | | VDD1 | V |
| V _{OS} | Input offset voltage | | –1.5 | ±0.2 | +1.5 | mV |
| TCV _{OS} | Input offset thermal drift | | –10 | ±1.5 | +10 | μV/K |
| CMRR | Common-mode rejection ratio | V _{IN} from 0 V to 5 V at 0 Hz | 108 | | | dB |
| | | V _{IN} from 0 V to 5 V at 50 kHz | 95 | | | dB |
| C _{IN} | Input capacitance to GND1 | VINP or VINN | 3 | | | pF |
| C _{IND} | Differential input capacitance | | 3.6 | | | pF |
| R _{IN} | Differential input resistance | | 28 | | | kΩ |
| | Small-signal bandwidth | | 60 | 100 | | kHz |
| OUTPUT | | | | | | |
| | Nominal gain | | 8 | | | |
| G _{ERR} | Gain error | Initial, at T _A = +25 °C | –0.5 | ±0.05 | +0.5 | % |
| | | | –1 | ±0.05 | +1 | % |
| TCG _{ERR} | Gain error thermal drift | | ±56 | | | ppm/K |
| | Nonlinearity | 4.5 V ≤ VDD2 ≤ 5.5 V | –0.075 | ±0.015 | +0.075 | % |
| | | 2.7 V ≤ VDD2 ≤ 3.6 V | –0.1 | ±0.023 | +0.1 | % |
| | Nonlinearity thermal drift | | 2.4 | | | ppm/K |
| | Output noise | VINP = VINN = 0 V | 3.1 | | | mV _{RMS} |
| PSRR | Power-supply rejection ratio | vs VDD1, 10-kHz ripple | 80 | | | dB |
| | | vs VDD2, 10-kHz ripple | 61 | | | dB |



AMC1200
AMC1200B

www.ti.com

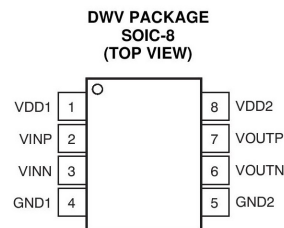
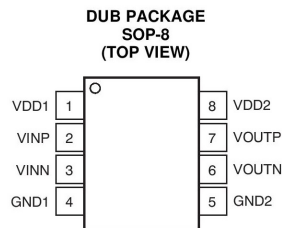
SBAS542C – APRIL 2011 – REVISED SEPTEMBER 2013

ELECTRICAL CHARACTERISTICS (continued)

All minimum/maximum specifications at $T_A = -40^\circ\text{C}$ to $+105^\circ\text{C}$ and within the specified voltage range, unless otherwise noted. Typical values are at $T_A = +25^\circ\text{C}$, $V_{DD1} = 5\text{ V}$, and $V_{DD2} = 3.3\text{ V}$.

| PARAMETER | TEST CONDITIONS | AMC1200, AMC1200B | | | UNIT |
|---------------------------------------|---|-------------------|------|------|-------------------------|
| | | MIN | TYP | MAX | |
| Rise/fall time | 0.5-V step, 10% to 90% | | 3.66 | 6.6 | μs |
| V_{IN} to V_{OUT} signal delay | 0.5-V step, 50% to 10%, unfiltered output | | 1.6 | 3.3 | μs |
| | 0.5-V step, 50% to 50%, unfiltered output | | 3.15 | 5.6 | μs |
| | 0.5-V step, 50% to 90%, unfiltered output | | 5.26 | 9.9 | μs |
| CMTI Common-mode transient immunity | $V_{CM} = 1\text{ kV}$ | 10 | 15 | | $\text{kV}/\mu\text{s}$ |
| Output common-mode voltage | $2.7\text{ V} \leq V_{DD2} \leq 3.6\text{ V}$ | 1.15 | 1.29 | 1.45 | V |
| | $4.5\text{ V} \leq V_{DD2} \leq 5.5\text{ V}$ | 2.4 | 2.55 | 2.7 | V |
| Short-circuit current | | | 20 | | mA |
| R_{OUT} Output resistance | | | 2.5 | | Ω |
| POWER SUPPLY | | | | | |
| V_{DD1} High-side supply voltage | | 4.5 | 5.0 | 5.5 | V |
| V_{DD2} Low-side supply voltage | | 2.7 | 5.0 | 5.5 | V |
| I_{DD1} High-side supply current | | | 5.4 | 8 | mA |
| I_{DD2} Low-side supply current | $2.7\text{ V} < V_{DD2} < 3.6\text{ V}$ | | 3.8 | 6 | mA |
| | $4.5\text{ V} < V_{DD2} < 5.5\text{ V}$ | | 4.4 | 7 | mA |
| P_{DD1} High-side power dissipation | | | 27.0 | 44.0 | mW |
| P_{DD2} Low-side power dissipation | $2.7\text{ V} < V_{DD2} < 3.6\text{ V}$ | | 11.4 | 21.6 | mW |
| | $4.5\text{ V} < V_{DD2} < 5.5\text{ V}$ | | 22.0 | 38.5 | mW |

PIN CONFIGURATIONS



PIN DESCRIPTIONS

| PIN # | PIN NAME | FUNCTION | DESCRIPTION |
|-------|----------|---------------|----------------------------|
| 1 | VDD1 | Power | High-side power supply |
| 2 | VINP | Analog input | Noninverting analog input |
| 3 | VINN | Analog input | Inverting analog input |
| 4 | GND1 | Power | High-side analog ground |
| 5 | GND2 | Power | Low-side analog ground |
| 6 | VOUTN | Analog output | Inverting analog output |
| 7 | VOUTP | Analog output | Noninverting analog output |
| 8 | VDD2 | Power | Low-side power supply |

B Excerto da ficha de dados XPT2046



XPT2046 Touch Screen Controller

Block Diagram

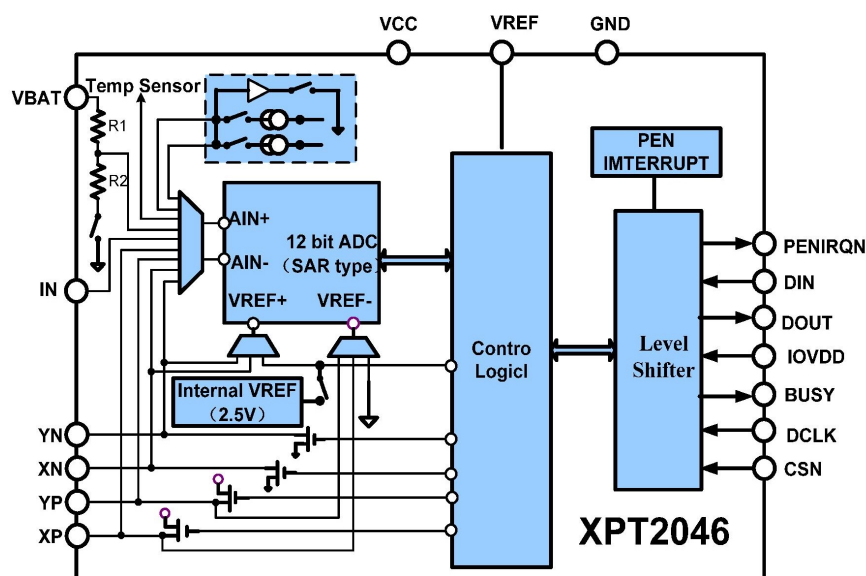


Figure 1. Block Diagram

Absolute Maximum Ratings

| | |
|-----------------------------------|-----------------------|
| +VCC and IOVDD to GND | -0.3V to +6V |
| Analog Inputs to GND | -0.3V to +VCC + 0.3V |
| Digital Inputs to GND | -0.3V to IOVDD + 0.3V |
| Power Dissipation | . 250mW |
| Maximum Junction Temperature | +150°C |
| Operating Temperature Range | . -40°C to +85°C |
| Storage Temperature Range | -65°C to +150°C |
| Lead Temperature (soldering, 10s) | +300°C |

Table 1. Absolute Maximum Ratings

WARNING: Stresses above these ratings may cause permanent damage. Exposure to absolute maximum conditions for xtended periods may degrade device reliability. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those specified is not implied.



XPT2046 Touch Screen Controller

Theory Of Operation

The XPT2046 is a classic successive approximation register (SAR) analog-to-digital converter (ADC). The architecture is based on capacitive redistribution, which inherently includes a sample-and-hold function. The converter is fabricated on a 0.6 μ m CMOS process. The basic operation of the XPT2046 is shown in Figure 4. The device features an internal 2.5V reference and uses an external clock. Operation is maintained from a single supply of 2.7V to 5.25V. The internal reference can be overdriven with an external, low-impedance source between 1V and +VCC. The value of the reference voltage directly sets the input range of the converter. The analog input (X-, Y-, and Z-Position coordinates, auxiliary input, battery voltage, and chip temperature) to the converter is provided via a multiplexer. A unique configuration of low on-resistance touch panel driver switches allows an unselected ADC input channel to provide power and the accompanying pin to provide ground for an external device, such as a touch screen. By maintaining a differential input to the converter and a differential reference architecture, it is possible to negate the error from each touch panel driver switch's on-resistance (if this is a source of error for the particular measurement).

Basic Operation of the XPT2046

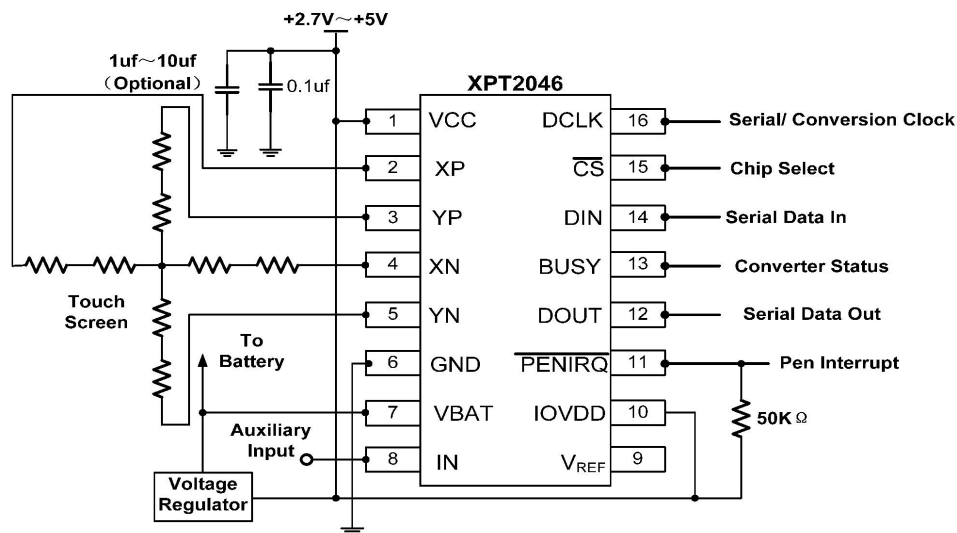


Figure 4. Basic Operation